Web graph mining

Prof. Michalis Vazirgiannis

LIX, Ecole Polytechnique,
&
INFRES , Telecom Paristech

http://www.lix.polytechnique.fr/~mvazirg/

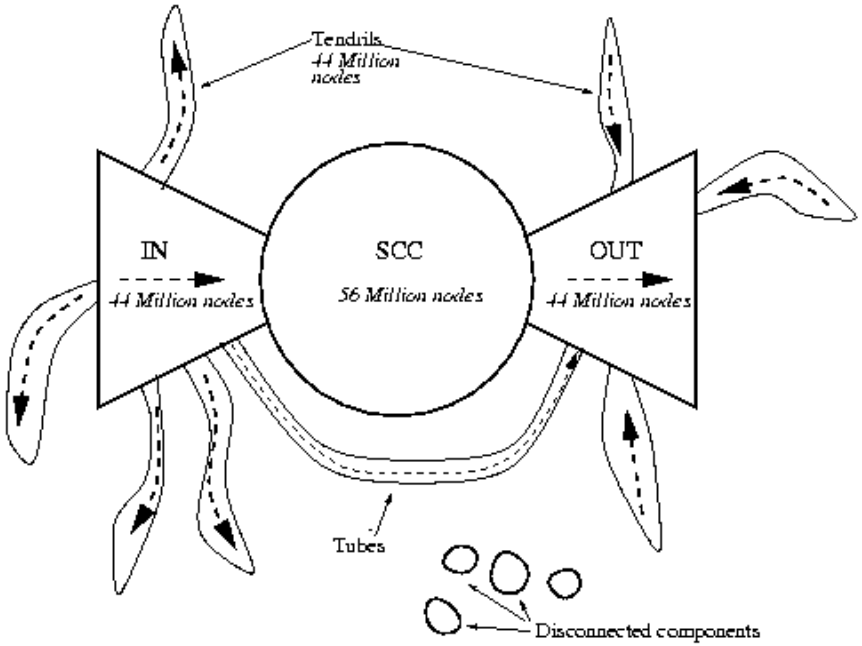
March 2012

Tutorial Outline

- Introduction, Motivation
- Data Preprocessing
  - Content preprocessing
  - Dimensionality reduction algorithms
- Ranking in the context of the Web graph
  - Graph Based ranking
    - Pagerank
    - HITS
  - Pagerank Computation methods
- Graph clustering for Community detection and evaluation
  - Graph based Clustering / spectral clustering
  - K-core, D-core structures and related measures
  - Case studies: DBLP, Wikipedia, Epinions

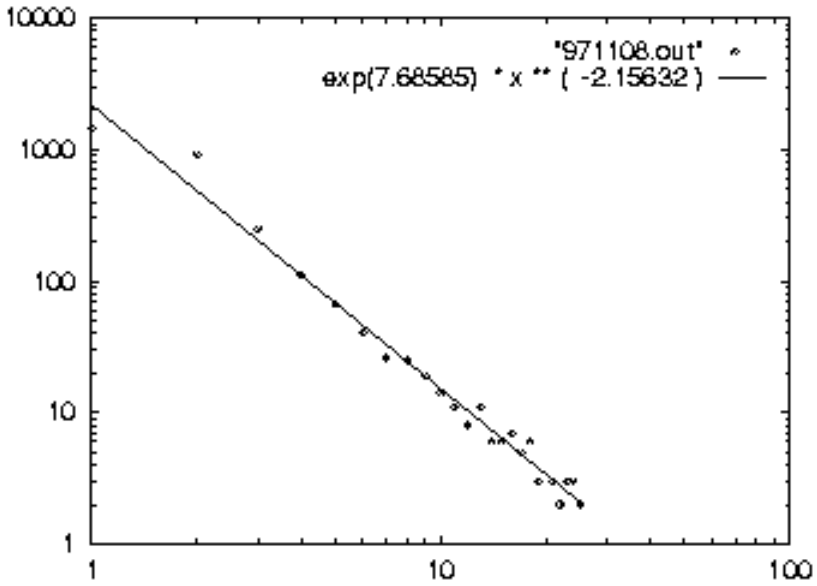
Web Mining - Introduction

- The Web looks like a “bow-tie” [Kumar+1999]
- IN, SCC, OUT, ‘tendrils’
- Disconnected components



- Power laws apply for count vs. (in-outlinks, user clicks)
- The plot is linear in log-log scale

freq = degree *(-2.15)*



Web Mining - I

The three aspects

- Web Content/ Structure / Usage Mining

Web content Mining

- Metadata: *Keywords, Phrases, Filename, URL, Size etc.*
- Feature extraction techniques from web contents
- Web document representation models: *Vector space model, ontology based ones*
- Web document clustering: *Similarity measures, Web document clustering algorithms*
- Web document classification techniques
- Content based ranking web documents – for queries

Web Mining - II

Web Structure Mining

- Link analysis theory (Hubs, Authorities, Co-citations, …)
- Mining web communities
- The role of hyperlinks in web crawling
- *Ranking* based on link analysis: *HITS, PageRank*

Web Usage Mining

- Web usage logs – click streams
- Data Preprocessing: *cleaning*, *session identification, User identification (anonymous user profiles vs. registered users)*
- Pattern discovery: Web usage-mining algorithms: *Classification, Clustering, Association Rules, Sequential pattern discovery*

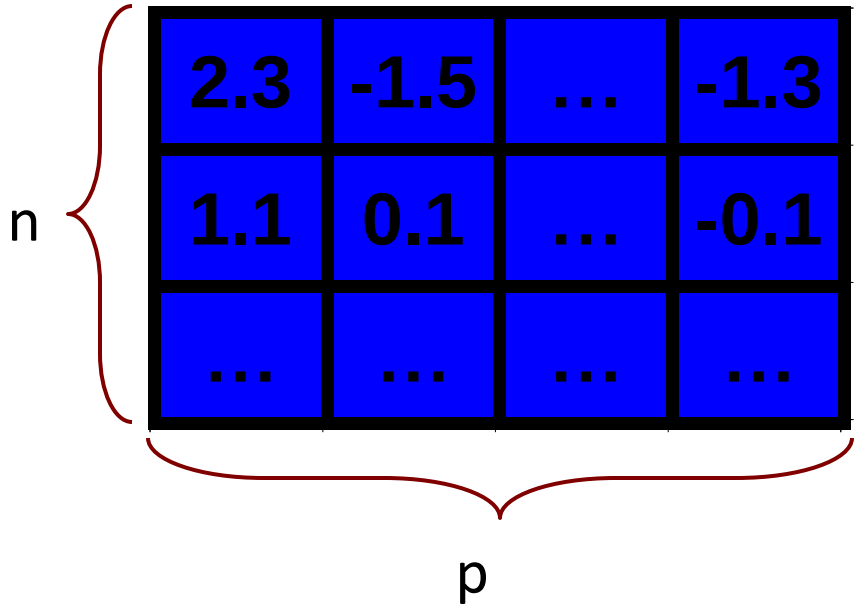
Web Mining - III

- Graph of web pages (mainly text)
- Huge: Google (~15 bn pages < 2% of the real web volume)
- Rapidly evolving: each week +8% pages, 25% of the links change!!
- Web search is a dominant activity! (Google as verb in dictionaries…)
- *Ranking* is an essential add-on towards meaningful results
- Further knowledge extraction and web organization is necessary

Tutorial Outline

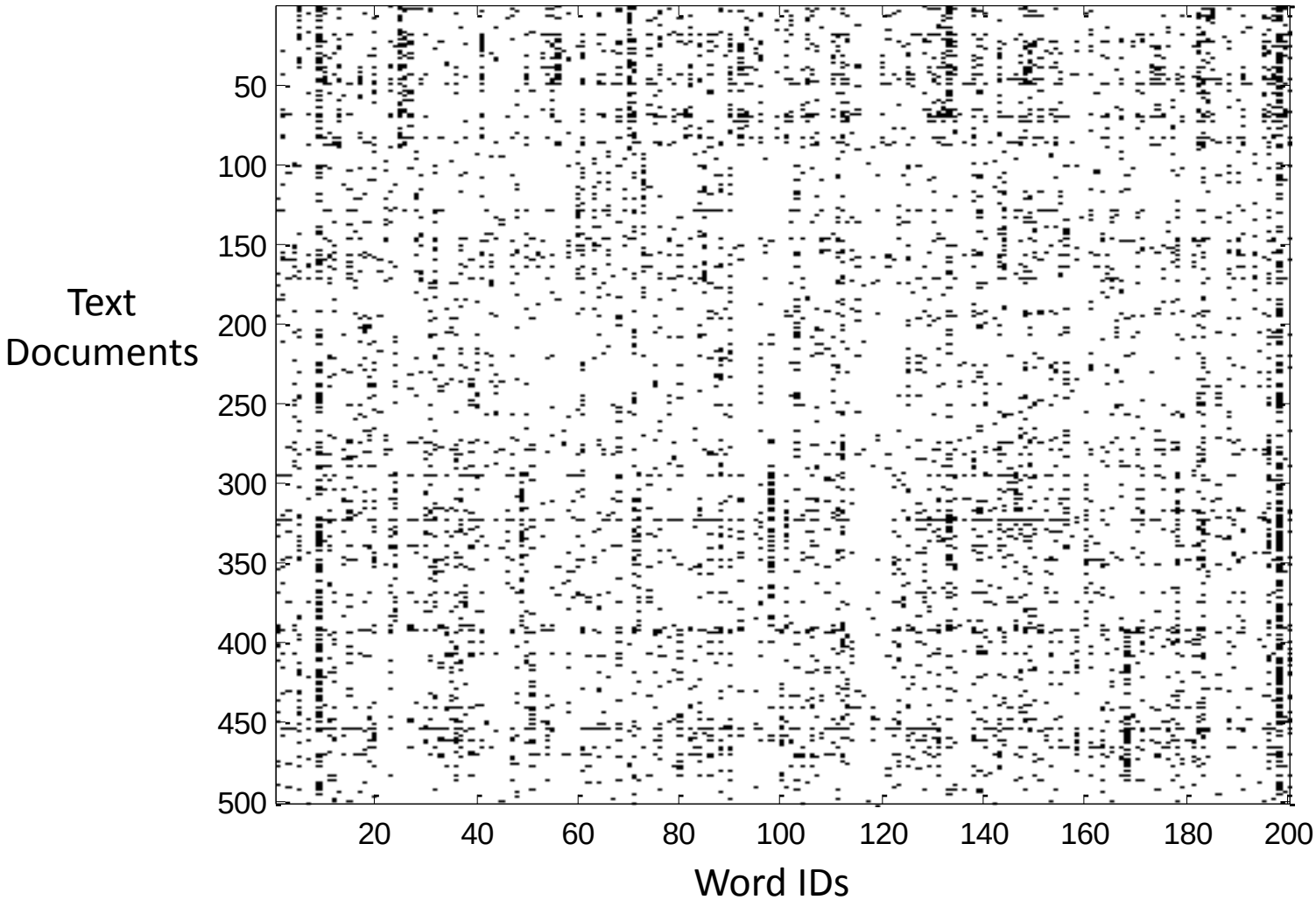
- Introduction, Motivation
- **Data Preprocessing**
  - Content preprocessing
  - Dimensionality reduction algorithms
- Ranking in the context of the Web graph
  - Graph Based ranking
    - Pagerank
    - HITS
  - Pagerank Computation methods
- Graph clustering for Community detection and evaluation
  - Graph based Clustering / spectral clustering
  - K-core, D-core structures and related measures
  - Case studies: DBLP, Wikipedia, Epinions

Content preprocessing



- Rows = objects
- Columns = measurements on objects
  - Represent each row as a p-dimensional vector, where p is the dimensionality
    - In efffect, embed our objects in a p-dimensional vector space
    - Often useful, but always appropriate
- Both n and p can be very large in certain data mining applications

Sparse Matrix (Text) Data



Tutorial Outline

- Introduction, Motivation
- **Data Preprocessing**
  - Content preprocessing
  - **Dimensionality reduction algorithms**
- Ranking in the context of the Web graph
  - Graph Based ranking
    - Pagerank
    - HITS
  - Pagerank Computation methods
- Graph clustering for Community detection and evaluation
  - Graph based Clustering / spectral clustering
  - K-core, D-core structures and related measures
  - Case studies: DBLP, Wikipedia, Epinions

Dim. Reduction – Linear Algorithms

- Principal Components Analysis (PCA)
- Singular Value Decomposition (SVD)
- Multidimensional Scaling (MDS)
- Latent Semantic Indexing (LSI)

Linear Algebra

Basic Principles

Dim. Reduction–Eigenvectors

A nxn matrix (square)

- eigenvalues λ: $|A-\lambda I|=0$
- Eigenvectors x : $Ax=\lambda x$
- Matrix rank: # linearly independent rows or columns
- A real symmetric table A nxn can be expressed as: $A=U\Lambda U^T$
- U's columns are A's eigenvectors
- Λ' s diagonal contains A's eigenvalues
- $A=U\Lambda U^T=\lambda_1 x_1 x^T_1+\lambda_2 x_2 x^T_2+\ldots+\lambda_n x_n x^T_n$
- $x_1 x^T_1$ represents projection via x_1 (λ_i eigenvalue, x_i eigenvector)

Linear Independence

- Linear Independence:
  - Given vectors $x_1^{\rightarrow}, x_2^{\rightarrow}, \ldots, x_n^{\rightarrow}$ and equation $\lambda_1 x_1^{\rightarrow} + \lambda_2 x_2^{\rightarrow} + \ldots + \lambda_n x_n^{\rightarrow} = 0$, if the equation has one solution, $\lambda_i = 0$, then all vectors are linear independent. If there are more solutions then the vectors are linearly dependent.
    - Linear dependence: One of the vectors is the result of the linear combination of one or more of remaining vectors.
- Basis of vector space V:
  - The set of linear independent vectors from which all elements of vector space V are produced as their linear combinations.
    - i.e. $B = \{e_1, e_2, \ldots, e_n\}$ with $e_i = \{0,0,0,\ldots,1,\ldots,0\}$ then B is the basis of R^n.
    - The same stands for $e_i = \{1,1,\ldots,1,0,\ldots,0\}$
- Dimension of vector space V:
  - The cardinality of its basis, dimV=n
    - Notice: A space V may have more than one basis. However, all basis are of the same cardinality.

Singular Value Decomposition (SVD) - I

Relation to eigenvectors/values

- Let **A mxn table**, can be expressed ULV^T
- **U:** mxm, its columns are $A*A^T$ eigenvectors.
- U,V define orthogonal basis: $UU^T = VV^T = 1$
- **L:** mxn contains A's singular values (square roots of $A*A^T$ eigenvalues)
- **V :** nxn, its columns are A^T*A eigenvectors

Singular Value Decomposition (SVD) - II

Matrix approximation

- The best rank r approximation M' of a matrix *M*. (minimizing the Frobenius norm)

$$\|A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2 = \operatorname{trace}(AA^H) = \sum_{i=1}^{\min\{m,n\}}\sigma_i^2$$

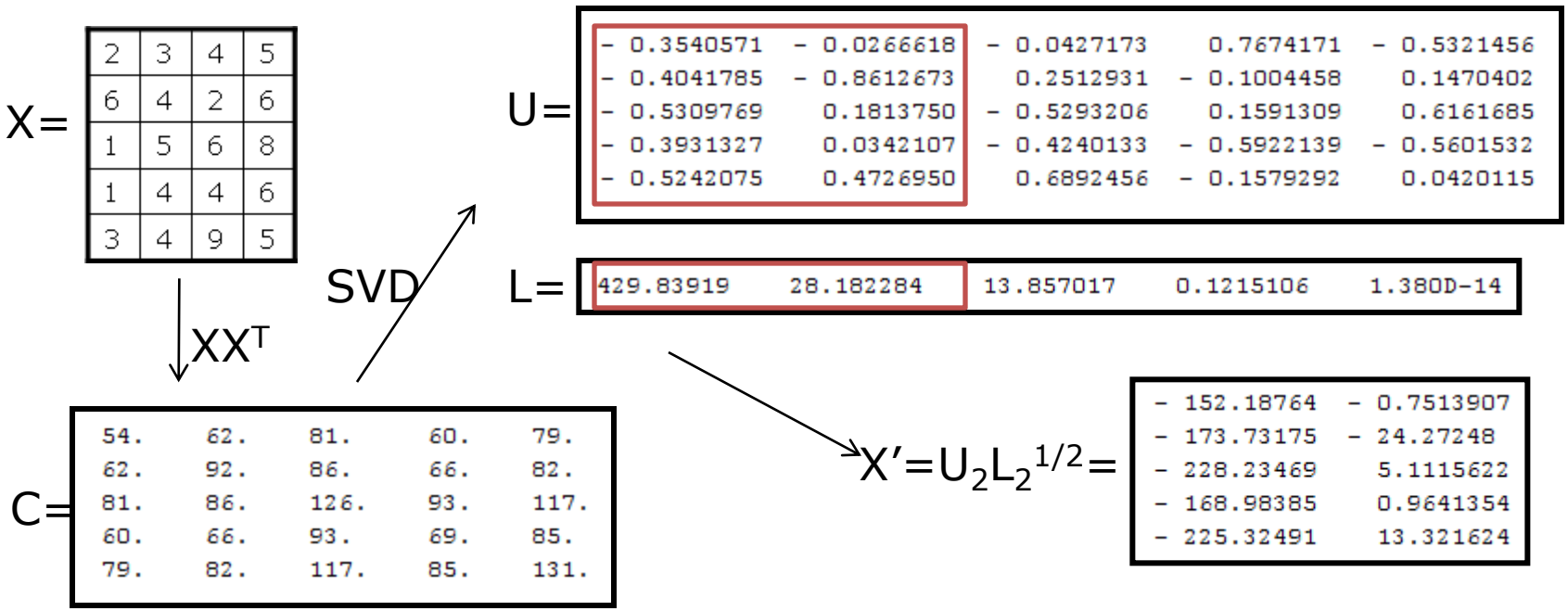
- M'=UΣ'V* (Σ' keeps the r largest singular values from Σ)
- Thus: distance ($||M||_F$, $||M'||_F$) is minimal

Multidimensional Scaling (MDS)

- Initially we depict vectors in random places
- Iteratively reposition them in order to minimize Stress.
  - **Stress =** $\sum(f(d_{ij})-d_{ij})^2/\sum f(d_{ij})^2$: The total error in distances mapping
  - Complexity $O(N^3)$ (N:number of vectors)
- Result:
  - Mapping of the data in a lower dimensional space.
- Implement usually by:
  - Eigen decomposition of the inner product matrix and projection on the k eigenvectors that correspond to the k largest eigenvalues.

Multidimensional Scaling

- Data is given as rows in X
  - $C=XX^T$ (inner product of x_i with x_j)
  - Eigen decomposition of $C' = ULU^{-1}$
  - Eventually $X' = U_k L_k^{1/2}$, where k is the projection dimension



Principal Components Analysis

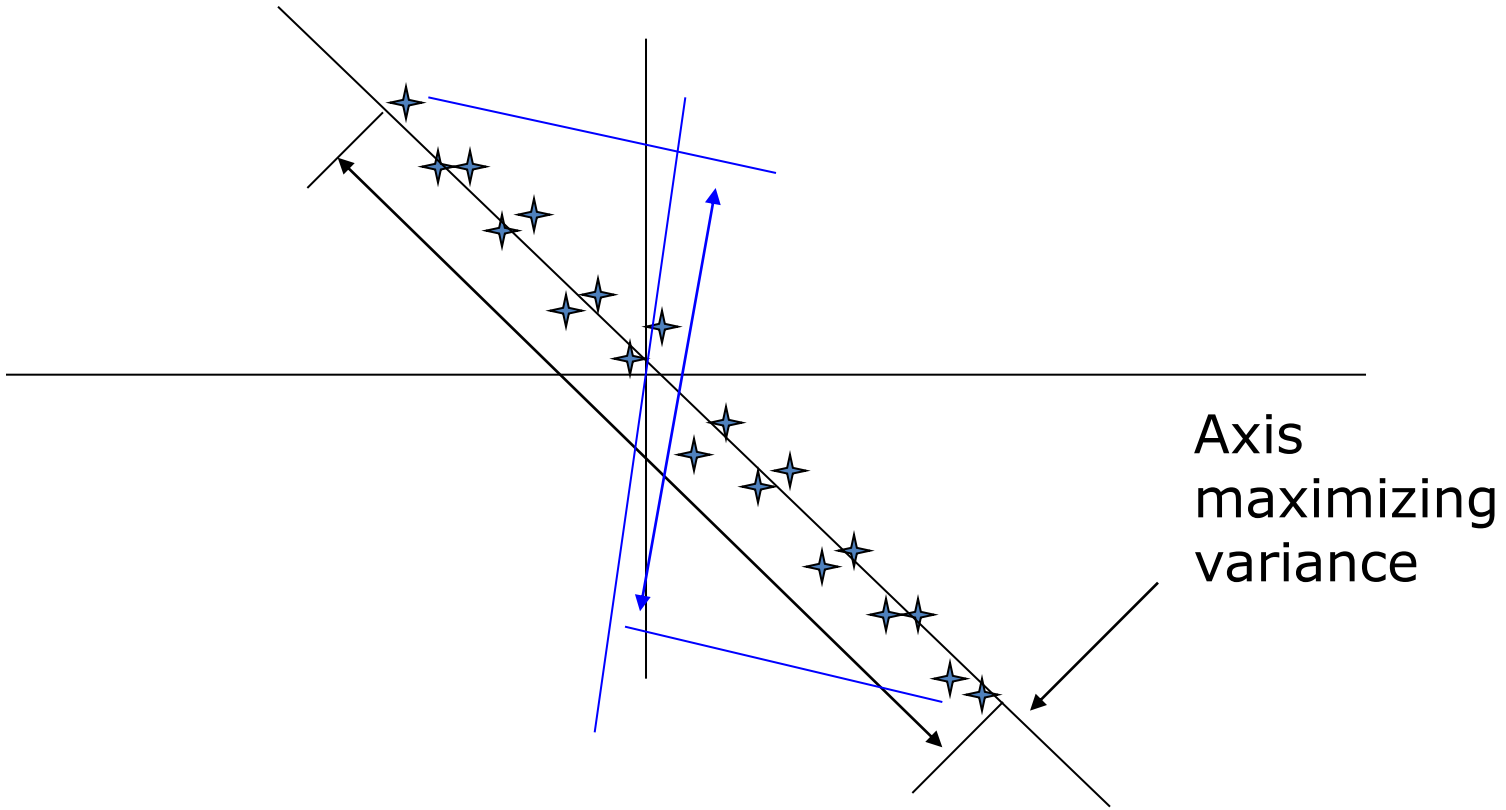
- The main concept behind *Principal Components Analysis* is dimensionality reduction, maintaining as much as possible data's variance.
- variance: $V(X)=\sigma^2=E[(X-\mu)^2]$
- Let *N* objects, with mean value, *m*, it is approximated as:

$$\frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2,$$

- In a sample of *N* objects with unknown mean value:

$$\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2,$$

Dimensionality reduction based on variance maintenance



Covariance Matrix

- Let Matrix $X = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$ where Xi vectors

- covariance matrix Σ is the matrix whose (*i*, *j*) entry is the covariance

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

- Also: cov(X) = XX^T

Principal Components Analysis (PCA)

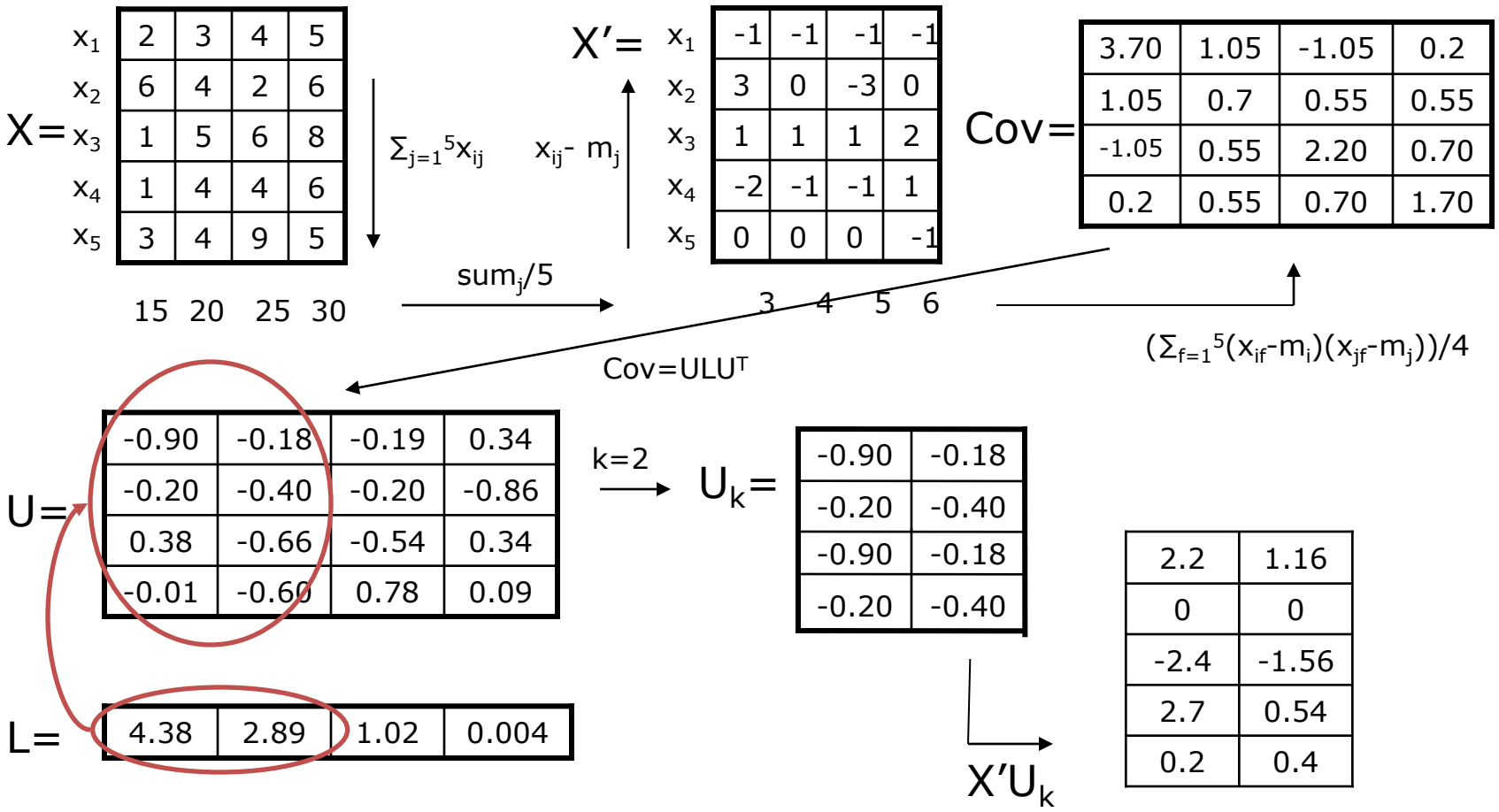
- The basic idea of PCA is the maximization of the covariance.
  - Variance: Depicts the maximum deviation of a random variable from the mean.
  - $\sigma^2=\sum_{i=1}^{n} ((x_i - \mu_i)^2/n)$

- Method:
  - Assumption: Data is described by p variables and contained as rows in matrix A_{pxn}
  - We subtract mean values from columns. A'=(A-M)
  - Calculate covariance matrix $W = A'^T A'$

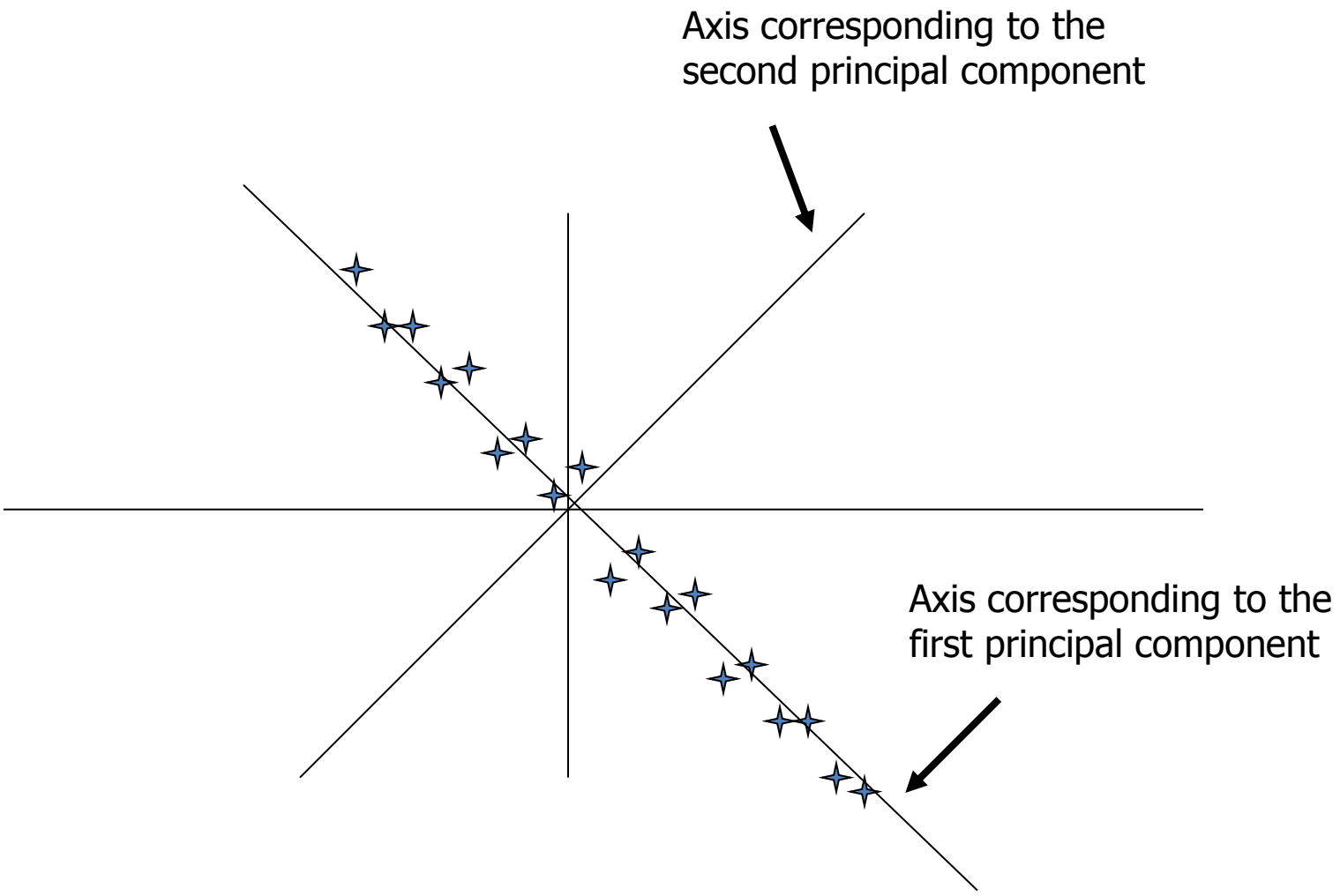
Principal Components Analysis (PCA) – (2)

- Calculation of covariance matrix (W)
  - A matrix nxn, in each cell of which (i,j) we have the covariance of X_i,X_j.
  - Covariance depicts the dependency between variables
    - If cov(X_i,X_j)=0 the X_i,X_j are independent
    - If cov(X_i,X_j)<0 then X_i,X_j are negative dependent (when i increases j decreases and vice versa)
    - If cov(X_i,X_j)>0 then X_i,X_j are positive dependent
    - Simply: $cov(X_i,X_j) = (\sum_{f=1}^{n}(X_{if} - \tilde{X_i})(X_{jf} - \tilde{X_j}))/(n-1)$
- Calculate eigenvalues and eigenvectors of W (X,D)
- Retain k largest eigenvalues and corresponding eigenvectors
  - k : input parameter such that $\sum_{j=k+1}^{p}\lambda_j / \sum_{j=1}^{p}\lambda_j >$ threshold
- Projection : $A'X_k$

Principal Components Analysis



PCA, example


Axis corresponding to the
second principal component
Axis corresponding to the
first principal component

PCA Applications

- It is a dimensionality reduction method
- Nominal complexity O(np^2+p^3)
  - n: number of data points
  - p: number of initial space dimensions
- The new space maintains sufficiently the data variance.
- Preprocessing step preceding the application of data mining algorithms (such as clustering).
- Data Visualization.
- Noise reduction.

Latent Structure in documents

- Documents are represented based on the Vector Space Model
- Vector space model consists of the keywords contained in a document.
- In many cases baseline keyword based performs poorly – not able to detect synonyms.
- Therefore document clustering is problematic
- Example where of keyword matching with the query: "IDF in computer-based information look-up"

	access	document	retrieval	information	theory	database	indexing	computer
Doc1	x	x	x			x	x	
Doc2				x	x			x
Doc3			x	x				x

Indexing by Latent Semantic Analysis (1990) Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Journal of the American Society of Information Science

Latent Semantic Indexing (LSI) - I

- Finding similarity with exact keyword matching is problematic.
- Using SVD we process the initial document-term document.
- Then we choose the k larger singular values. The resulting matrix is of order k and is the most similar to the original one based on the Frobenius norm than any other k-order matrix.

Latent Semantic Indexing (LSI) - II

- The initial matrix is SVD decomposed as: $A=ULV^T$
- Choosing the top-k singular values from L we have:

$$A_k=U_kL_kV_k^T ,$$

- L_k is square kxk containing the top-k singular values of the diagonal in matrix L,
- U_k, the mxk matrix containing the first k columns in U (left singular vectors)
- V_k^T, the kxn matrix containing the first k lines of V^T (right singular vectors)

Typical values for κ~200-300 (empirically chosen based on experiments appearing in the bibliography)

LSI capabilities

- Term to term similarity: $A_k A_k^T = U_k L_k^2 U_k^T$
- document-document similarity: $A_k^T A_k = V_k L_k^2 V_k^T$
- term document similarity (as an element of the transformed – document matrix)
- Extended query capabilities transforming initial query q to q_n : $q_n = q^T U_k L_k^{-1}$
- Thus q_n can be regarded a line in matrix V_k

LSI – an example [1]

LSI application on a term – document matrix

C1: Human machine Interface for Lab ABC computer application

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 measurements

M1: The generation of random, binary unordered trees

M2: The intersection graph of path in trees

M3: Graph minors IV: Widths of trees and well-quasi-ordering

M4: Graph minors: A survey

- The dataset consists of 2 classes, 1st: "human – computer interaction" (c1-c5) 2nd: related to graph (m1-m4). After feature extraction the titles are represented as follows.

---

[1] Indexing by Latent Semantic Analysis (1990) Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Journal of the American Society of Information Science

LSI – an example

	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

LSI – an example

$A = ULV^T$

$A =$

1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

LSI – an example

$A = ULV^T$

$U =$

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

LSI – an example

$A=ULV^T$

L=

3.34	0	0	0	0	0	0	0	0
0	2.54	0	0	0	0	0	0	0
0	0	2.35	0	0	0	0	0	0
0	0	0	1.64	0	0	0	0	0
0	0	0	0	1.50	0	0	0	0
0	0	0	0	0	1.31	0	0	0
0	0	0	0	0	0	0.85	0	0
0	0	0	0	0	0	0	0.56	0
0	0	0	0	0	0	0	0	0.36
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

LSI – an example

$A=ULV^T$

V=

0.20	-0.06	0.11	-0.95	0.05	-0.08	0.18	-0.01	-0.06
0.61	0.17	-0.50	-0.03	-0.21	-0.26	-0.43	0.05	0.24
0.46	-0.13	0.21	0.04	0.38	0.72	-0.24	0.01	0.02
0.54	-0.23	0.57	0.27	-0.21	-0.37	0.26	-0.02	-0.08
0.28	0.11	-0.51	0.15	0.33	0.03	0.67	-0.06	-0.26
0.00	0.19	0.10	0.02	0.39	-0.30	-0.34	0.45	-0.62
0.01	0.44	0.19	0.02	0.35	-0.21	-0.15	-0.76	0.02
0.02	0.62	0.25	0.01	0.15	0.00	0.25	0.45	0.52
0.08	0.53	0.08	-0.03	-0.60	0.36	0.04	-0.07	-0.45

LSI – an example

Choosing the 2 largest singular values we have

$U_k =$

0.22	-0.11
0.20	-0.07
0.24	0.04
0.40	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.30	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

$L_k =$

3.34	0
0	2.54

$V_k^T =$

0.20	0.61	0.46	0.54	0.28	0.00	0.02	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53

LSI (2 singular values)

A_k =

	C1	C2	C3	C4	C5	M1	M2	M3	M4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
Interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
Computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
User	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
System	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
Response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
Time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
Survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
Trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
Graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
Minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

LSI Example

- Query: "human computer interaction" retrieves documents: c_1,c_2, c_4 but *not* c_3 and c_5.
- If we submit the same query (based on the transformation shown before) to the transformed matrix we retrieve (using cosine similarity) all c_1-c_5 even if c_3 and c_5 have no common keyword to the query.
- According to the transformation for the queries we have:

Query transformation

	query
human	1
Interface	0
computer	1
User	0
System	0
Response	0
Time	0
EPS	0
Survey	0
Trees	0
Graph	0
Minors	0

q=

1
0
1
0
0
0
0
0
0
0
0
0

Query transformation

$q^T =$

1	0	1	0	0	0	0	0	0	0	0	0

$U_k =$

0.22	-0.11
0.20	-0.07
0.24	0.04
0.40	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.30	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

$L_k^{-1} =$

0.3	0
0	0.39

$q_n = q^T U_k L_k^{-1} =$

0.138	-0.0273

Query transformation

$V_k L_k =$

0.20	-0.06
0.61	0.17
0.46	-0.13
0.54	-0.23
0.28	0.11
0.00	0.19
0.01	0.44
0.02	0.62
0.08	0.53

3.34	0
0	2.54

=

0.67	-0.15
2.04	0.43
1.54	-0.33
1.80	-0.58
0.94	0.28
0.00	0.48
0.03	1.12
0.07	1.57
0.27	1.35

$q_n L_k =$

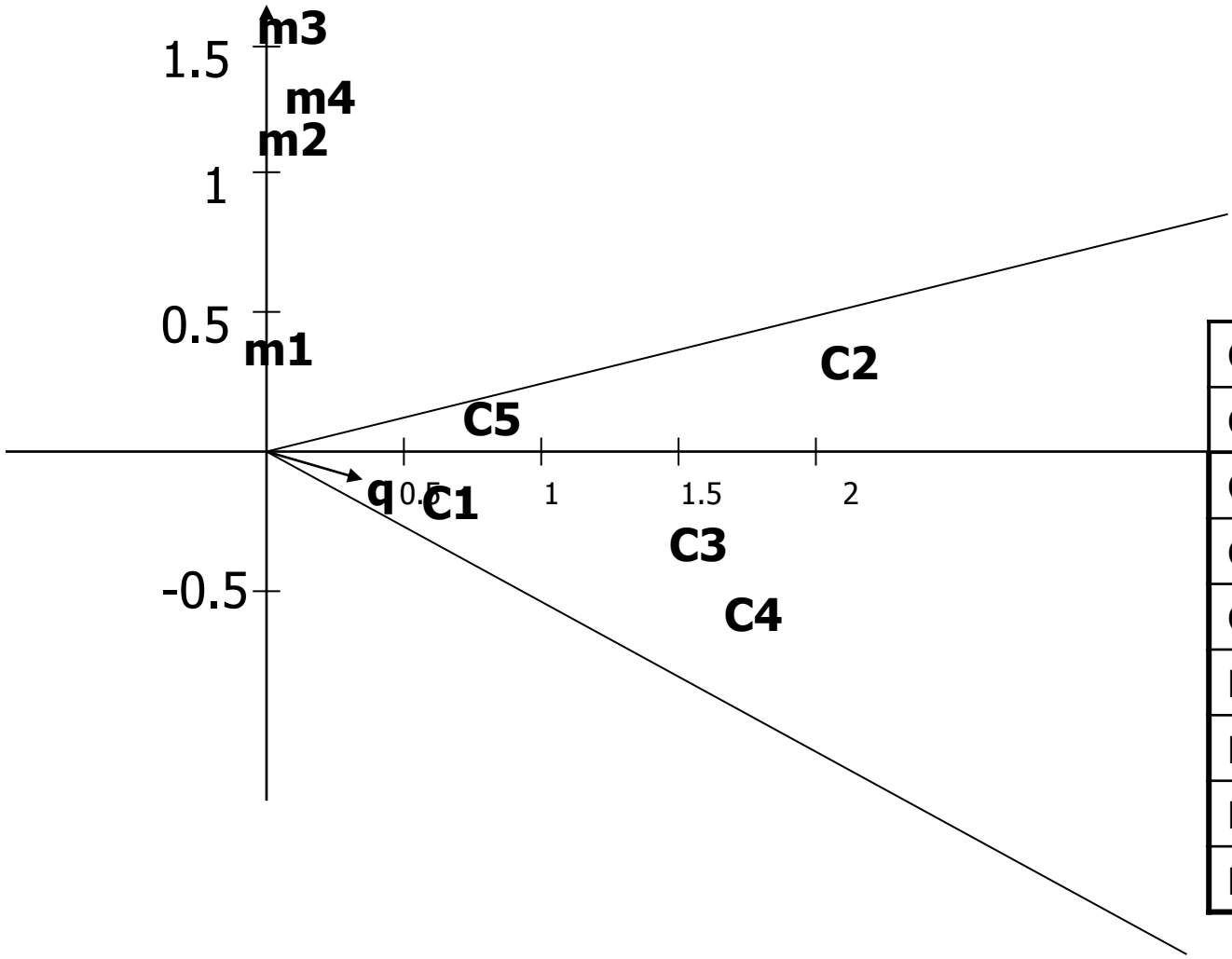
0.138	-0.0273

3.34	0
0	2.54

=

0.46	-0.069

Query document similarity in LSI space



	query
C1	0.99
C2	0.94
C3	0.99
C4	0.99
C5	0.90
M1	-0.14
M2	-0.13
M3	-0.11
M4	0.05

Tutorial Outline

- Introduction, Motivation
- Data Preprocessing
  - Content preprocessing
  - Dimensionality reduction algorithms
- **Ranking in the context of the Web graph**
  - Graph Based ranking
    - Pagerank
    - HITS
  - Pagerank Computation methods
- Graph clustering for Community detection and evaluation
  - Graph based Clustering / spectral clustering
  - K-core, D-core structures and related measures
  - Case studies: DBLP, Wikipedia, Epinions
  -

Outline

- Motivation
- *Short intro to Pagerank ?*
- Web Rank Change measures
- Markov Model Learning
- Page ranking Predictions
- Conclusions – Further Work

Why link analysis?

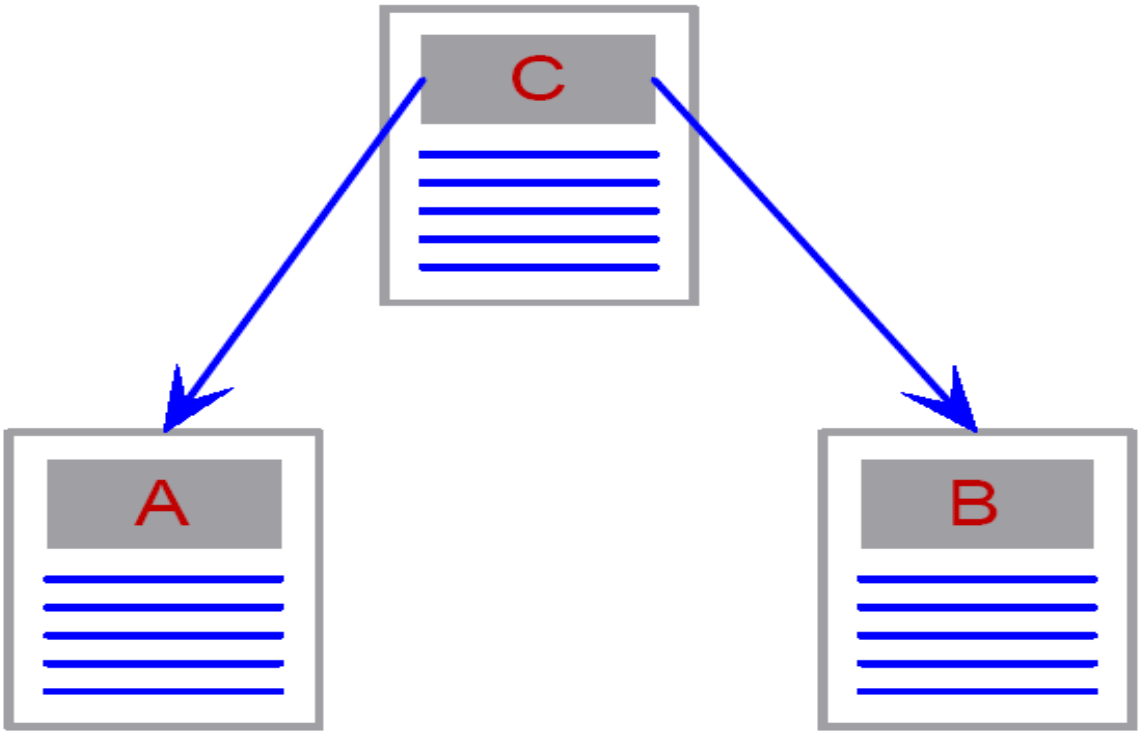
- The web is **not** just a collection of documents – its hyperlinks are important!
- A link from page *A* to page *B* may indicate:
  - *A* is related to *B*, or
  - *A* is recommending, citing or endorsing *B*
- Links are either
  - referential – *click here and get back home*, or
  - Informational – *click here to get more detail*

Citation Analysis

- The **impact factor** of a journal = *A/B*
  - *A* is the number of current year citations to articles appearing in the journal during previous two years.
  - *B* is the number of articles published in the journal during previous two years.

Journal Title	**Impact Factor (2002)**
J. Mach. Learn. Res.	3.818
IEEE T. Pattern Anal.	2.923
Mach. Learn.	1.944
IEEE Intell. Syst.	1.905
Artif. Intell.	1.703

Co-Citation



- *A* and *B* are co-cited by *C*, implying that
  - they are related or associated.
- The strength of co-citation between *A* and *B* is the number of times they are co-cited.

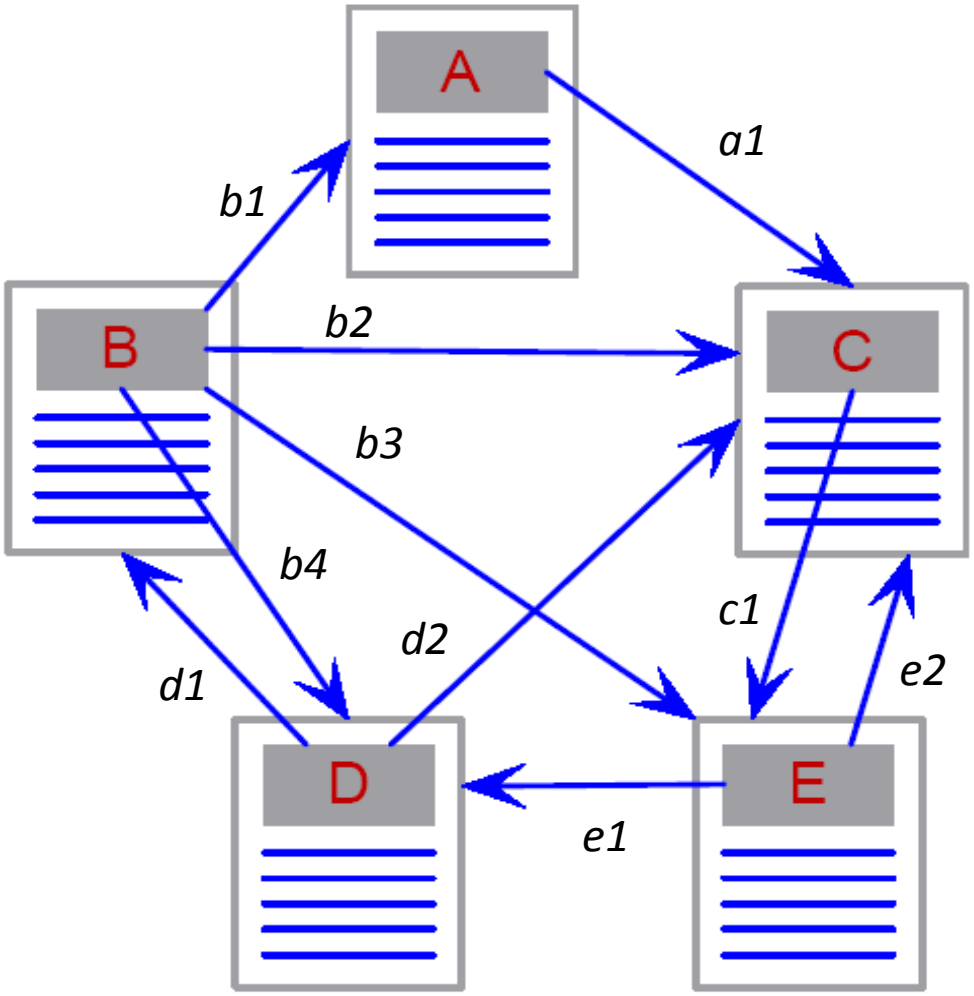
What is a Markov Chain?

A Markov chain has two components:

- A network structure much like a web site, where each node is called a state.
- A transition probability of traversing a link given that the chain is in a state.
  - For each state the sum of outgoing probabilities is one.

A sequence of steps through the chain is called a *random walk*.

Markov Chain Example



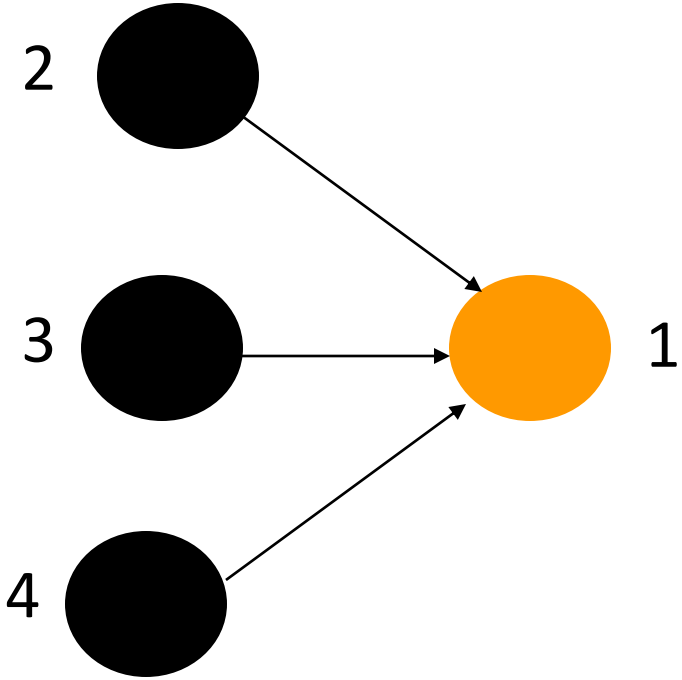
HITS - Kleinberg's Algorithm

- HITS – Hypertext Induced Topic Selection
- For each vertex v Є V in a subgraph of interest:

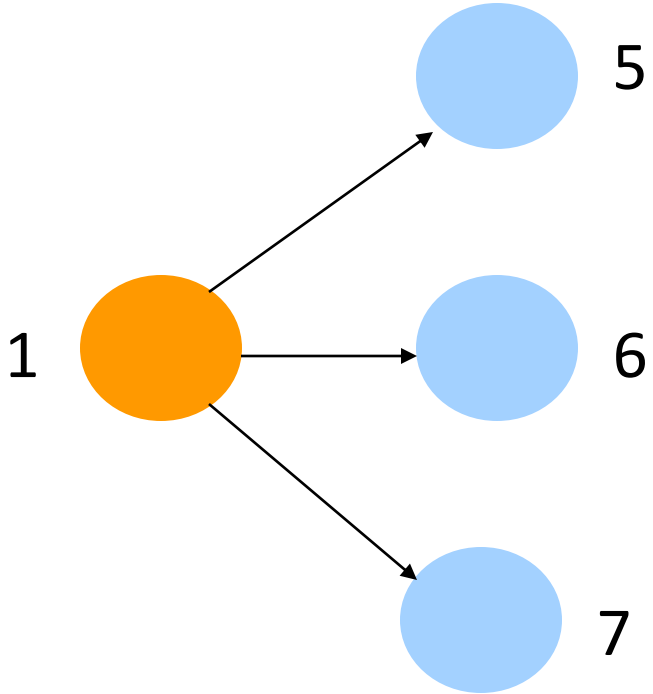
  a(v) - the authority of v
  h(v) - the hubness of v

- A site is very authoritative if it receives many citations. Citation from important sites weight more than citations from less-important sites
- Hubness shows the importance of a site. A good hub is a site that links to many authoritative sites

Authority and Hubness



a(1) = h(2) + h(3) + h(4)



h(1) = a(5) + a(6) + a(7)

Authority and Hubness Convergence

- Recursive dependency:

$$a(v) \leftarrow \sum_{w \in pa[v]} h(w)$$

$$h(v) \leftarrow \sum_{w \in ch[v]} a(w)$$

- Using Linear Algebra, we can prove:

a(v) and h(v) converge

HITS Example

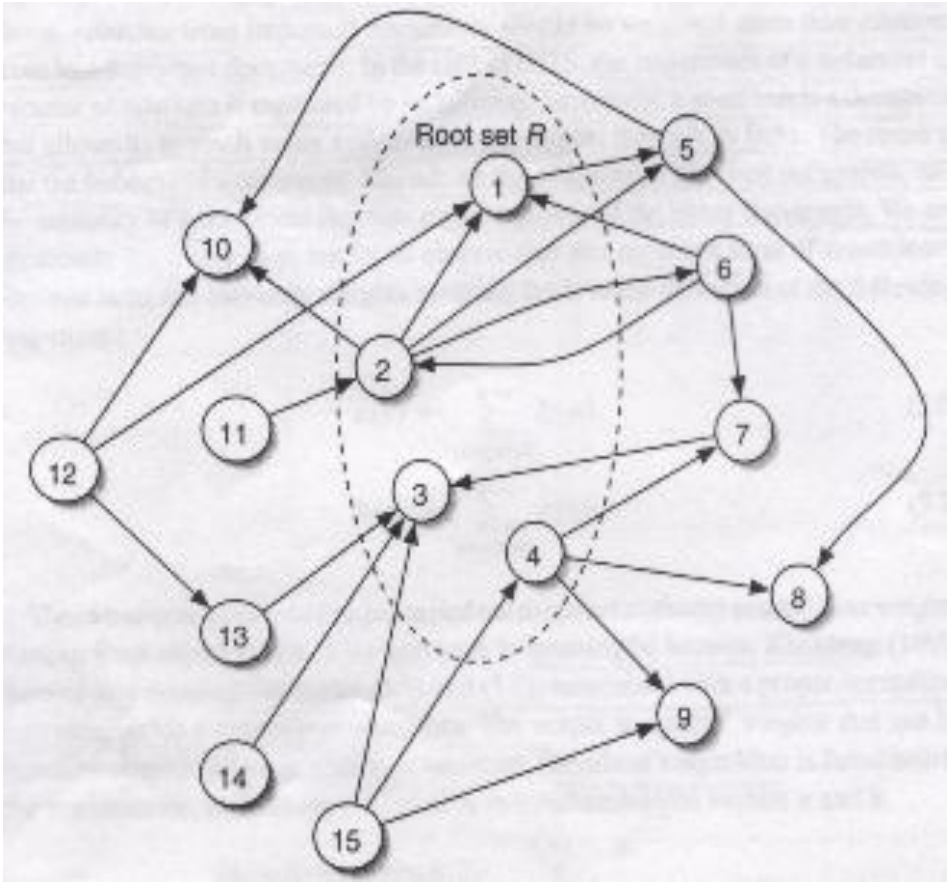
Find a base subgraph:

- Start with a root set R {1, 2, 3, 4}

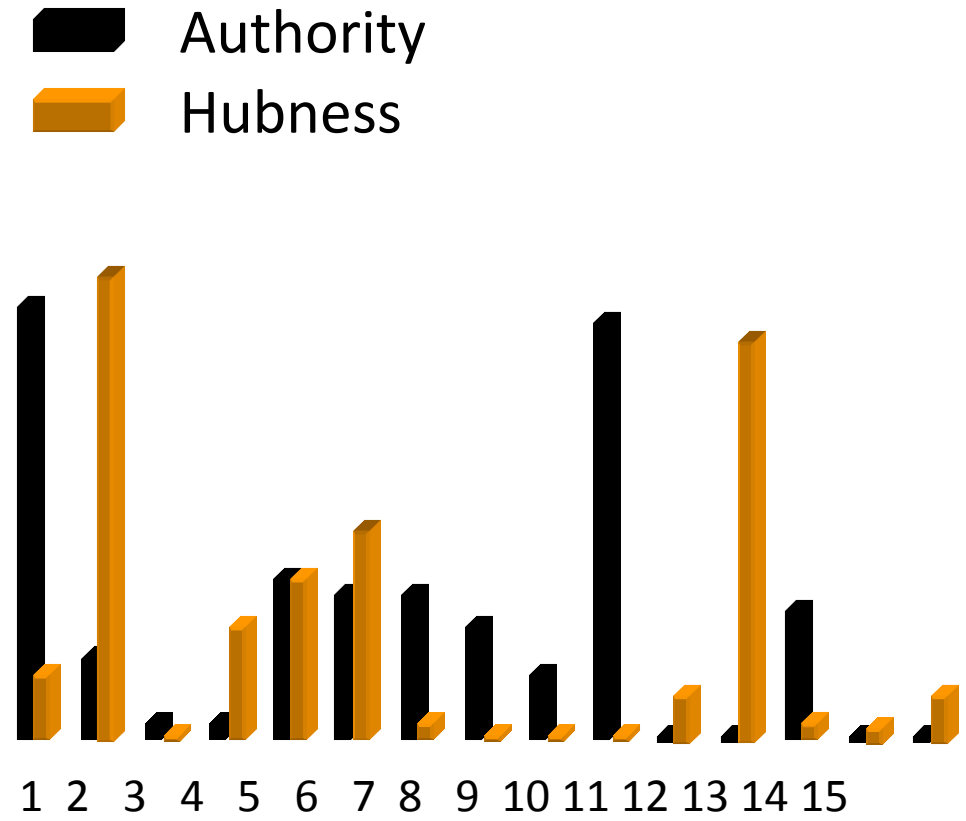
- {1, 2, 3, 4} - nodes relevant to the topic

- Expand the root set R to include all the children and a fixed number of parents of nodes in R

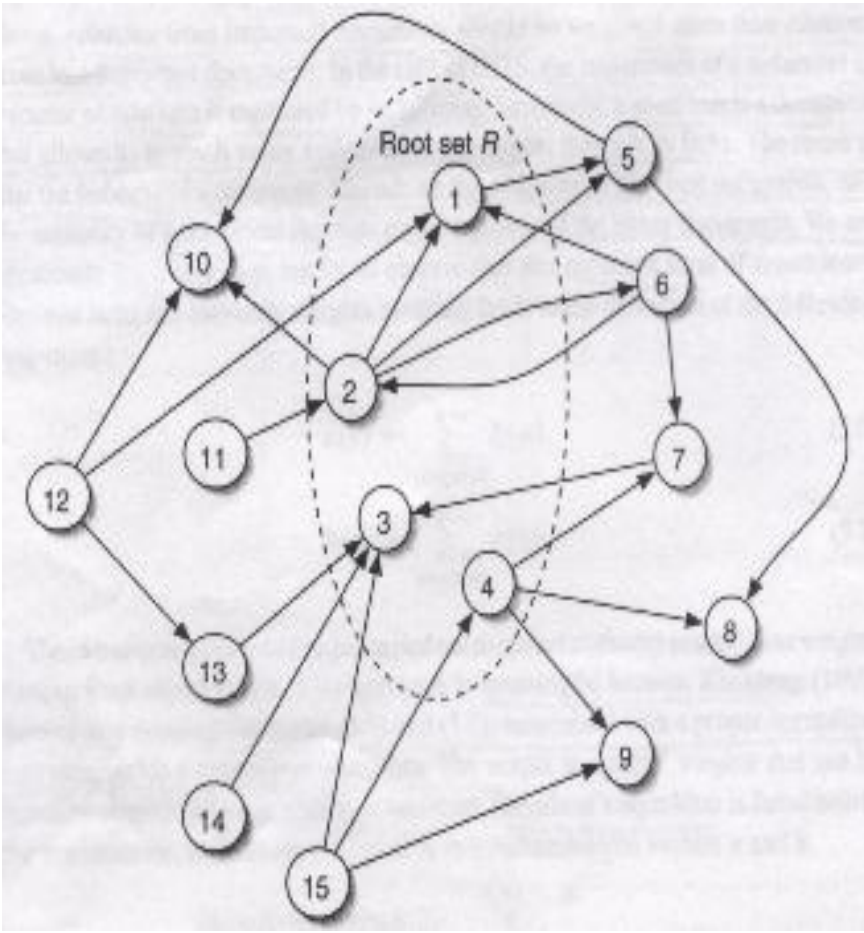
→ A new set S (base subgraph) →



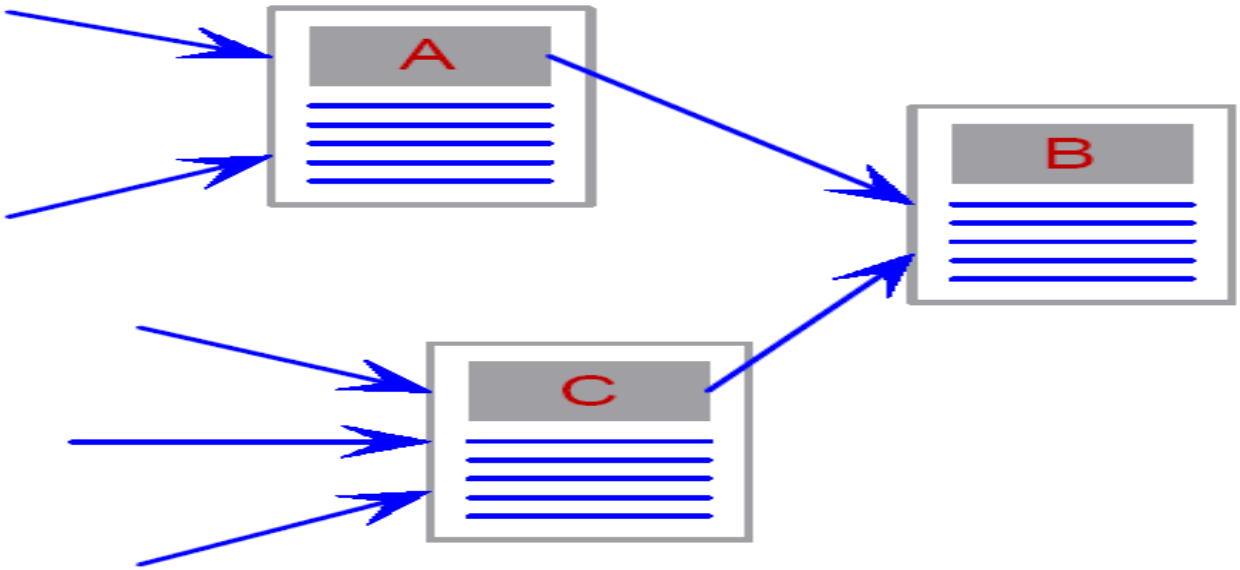
HITS Example Results



Authority and hubness weights



PageRank - Motivation

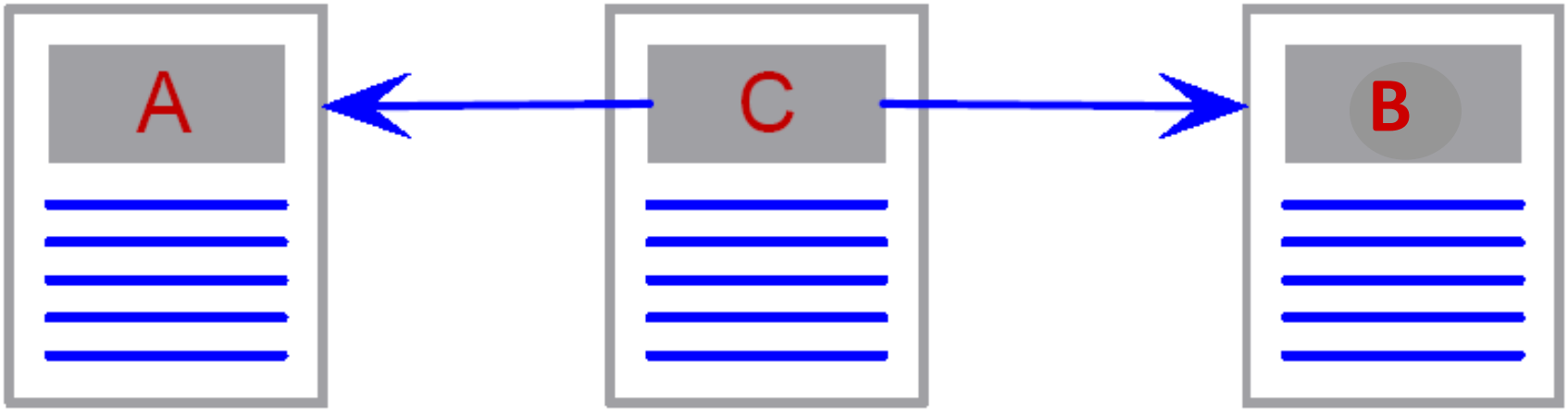


- A link from page *A* to page *B* is a **vote** of the author of *A* for *B,* or a **recommendation** of the page.
- The number incoming links to a page is a measure of importance and authority of the page.
- Also take into account the quality of recommendation, so a page is more important if the sources of its incoming links are important.

The Random Surfer

- Assume the web is a Markov chain.
- Surfers randomly click on links, where the probability of an outlink from page A is *1/m*, where *m* is the number of outlinks from *A*.
- The surfer occasionally gets *bored* and is *teleported* to another web page, say *B,* where *B* is equally likely to be any page.
- Using the theory of Markov chains it can be shown that if the surfer follows links for long enough, *the PageRank of a web page is the probability that the surfer will visit that page*.

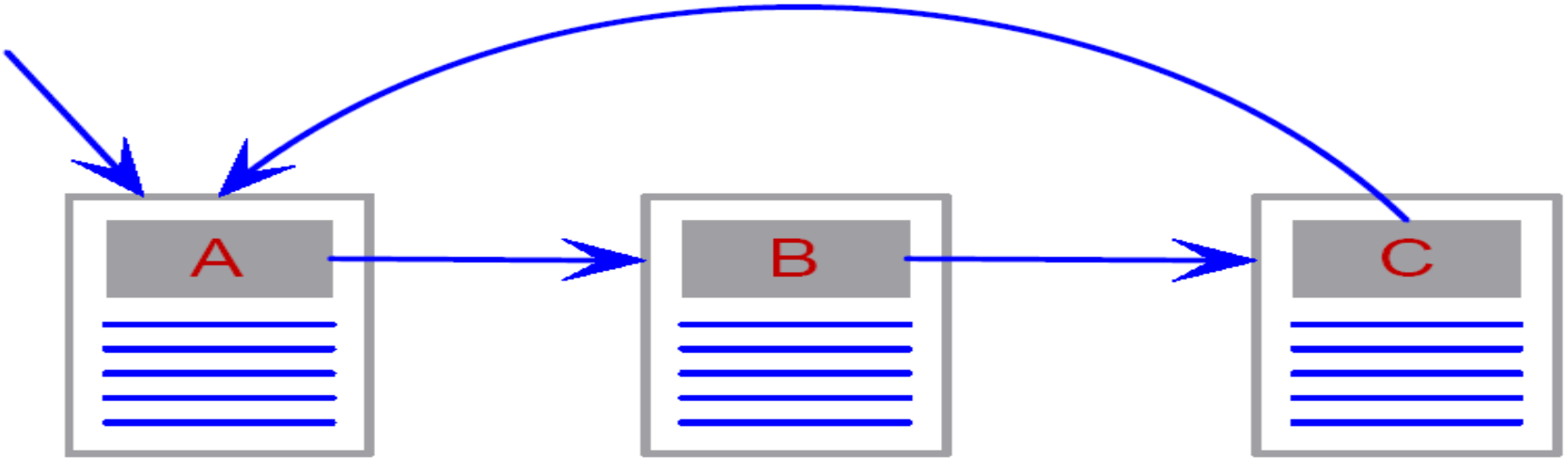
Dangling Pages



- Problem: *A* and *B* have no outlinks.

Solution: Assume *A* and *B* have links to all web pages with equal probability.

Rank Sink



- Problem: Pages in a loop accumulate rank but do not distribute it.
- Solution: Teleportation, i.e. with a certain probability the surfer can jump to any other web page to get out of the loop.

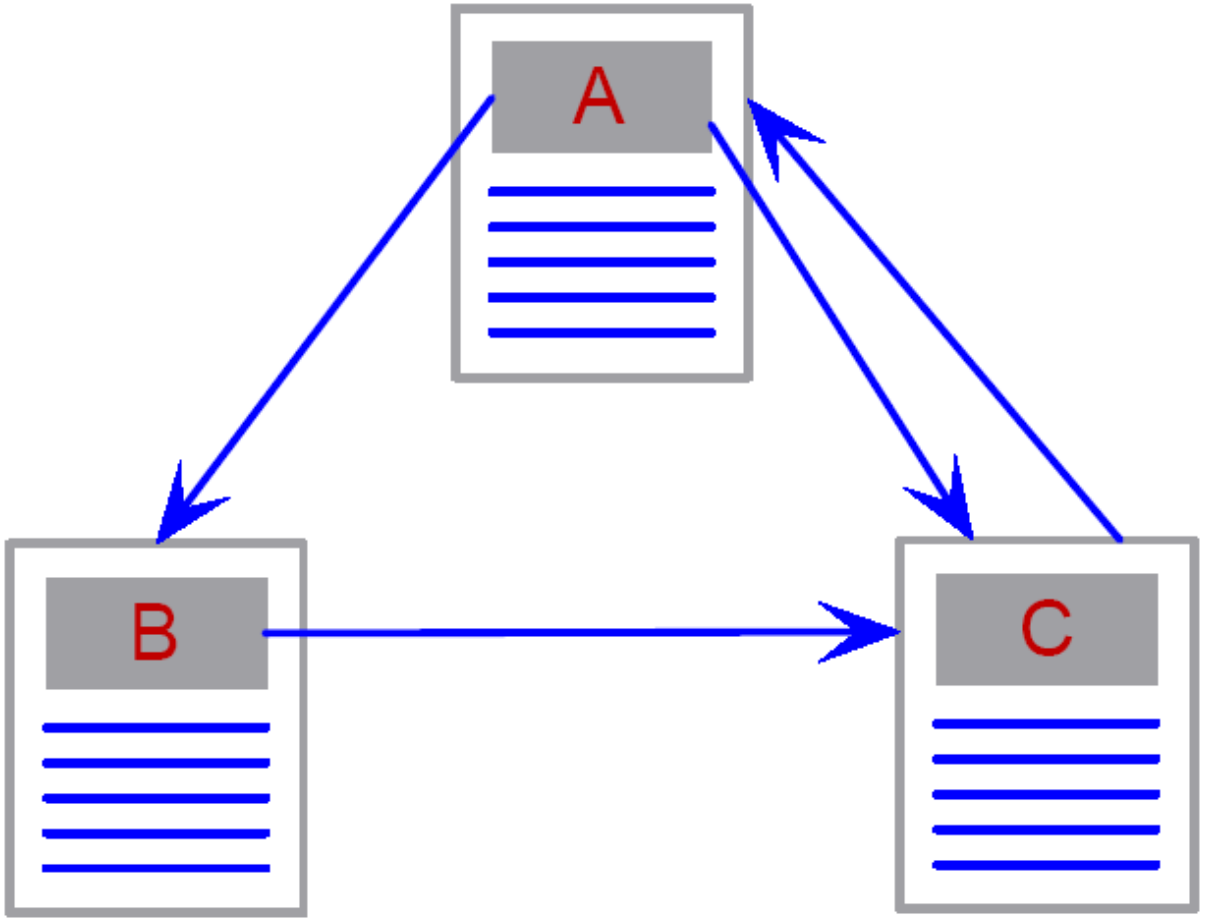
PageRank (*PR*) – Definition

$$PR(P) = \frac{d}{N} + (1-d)\left(\frac{PR(P_1)}{O(P_1)} + \frac{PR(P_2)}{O(P_2)} + \ ...+ \ \frac{PR(P_n)}{O(P_n)}\right)$$

- *P* is a web page
- *Pi* are the web pages that have a link *to P*
- *O*(*Pi*) is the number of outlinks from *Pi*
- *d* is the teleportation probability
- *N* is the size of the web

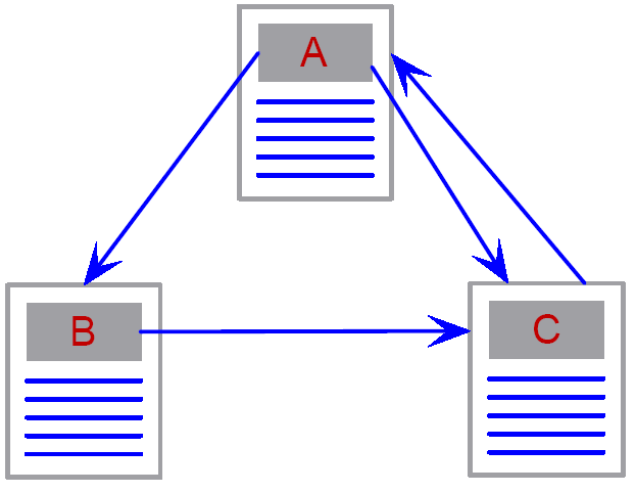
- Difference to HITS
  - HITS takes Hubness & Authority weights
  - The page rank is proportional to its parents' rank, but inversely proportional to its parents' outdegree

Example Web Graph



Iteratively Computing PageRank

- d is normally set to 0.15
- Set initial *PR* values to 1/3
- *Solve the following equations iteratively:*



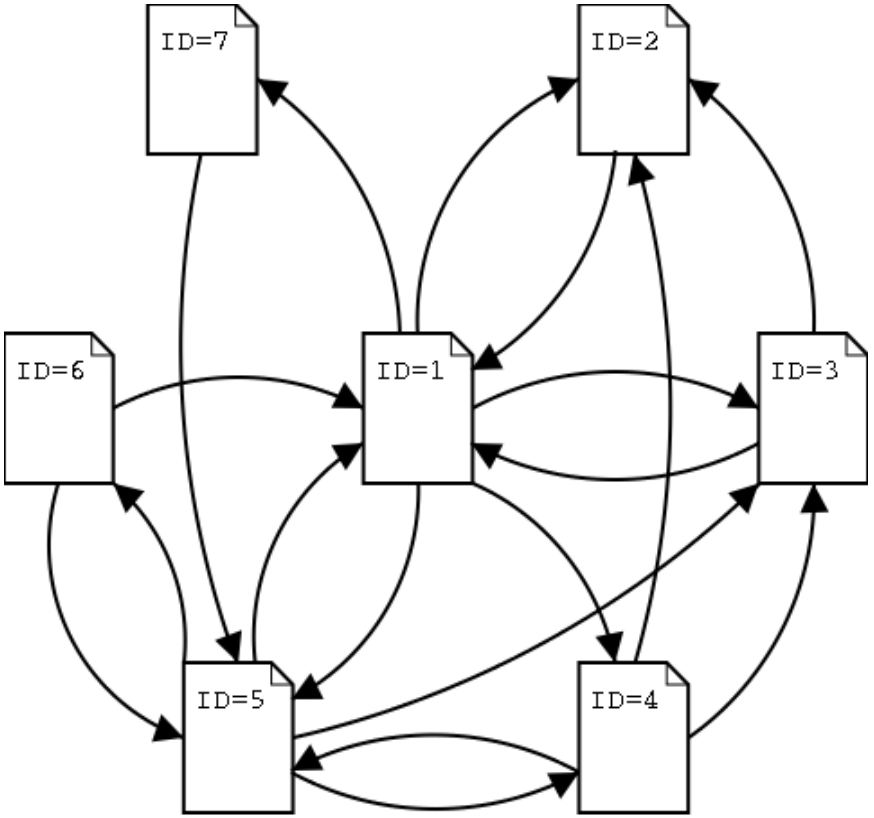
$$PR(A) = 0.15/3 + 0.85PR(C)$$
$$PR(B) = 0.15/3 + 0.85(PR(A)/2)$$
$$PR(C) = 0.15/3 + 0.85(PR(A)/2 + PR(B))$$

Example Computation of PR

	PR(A)	PR(B)	PR(C)	ERROR
1	0,333333333	0,333333333	0,333333333	
2	0,333333333	0,191666667	0,475	0,04014
3	0,45375	0,191666667	0,354583333	0,029
4	0,351395833	0,24284375	0,405760417	0,01571
5	0,394896354	0,199343229	0,405760417	0,00378
6	0,394896354	0,217830951	0,387272695	0,00068
7	0,379181791	0,217830951	0,402987258	0,00049
8	0,39253917	0,211152261	0,396308569	0,00027
9	0,386862284	0,216829147	0,396308569	6,4E-05
10	0,386862284	0,214416471	0,398721246	1,2E-05
11	0,388913059	0,214416471	0,396670471	8,4E-06
12	0,3871699	0,21528805	0,39754205	4,6E-06
13	0,387910742	0,214547208	0,39754205	1,1E-06
14	0,387910742	0,214862066	0,397227192	2E-07
15	0,387643113	0,214862066	0,397494821	1,4E-07
16	0,387870598	0,214748323	0,397381079	7,8E-08
17	0,387773917	0,214845004	**0,397381079**	1,9E-08

- Error converges fast, ~10 repetitions
- Page C is the top ranked one

Matrix Notation



Page ID	OutLinks
1	2,3,4,5,7
2	1
3	1,2
4	2.3.5
5	1,3,4,6
6	1.5
7	5

Adjacent Matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

* http://www.kusatro.kyoto-u.com

Matrix Notation

- Matrix Notation

  $\mathbf{r} = \alpha \mathbf{B}\,\mathbf{r} = \mathbf{M}\,\mathbf{r}$

  α : eigenvalue

  r : eigenvector of B

$$b_{uv} = \begin{cases} \dfrac{a_{uv}}{\sum_w a_{uw}} & \text{if } ch[u] \neq 0, \\ a_{uv} = 0 & \text{otherwise} \end{cases}$$

$$A\,x = \lambda\,x$$

$$|\,A - \lambda I\,|\,x = 0$$

$$B = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Finding Pagerank

→ to find eigenvector of B with an associated eigenvalue α

Matrix Notation

PageRank: eigenvector of **P** relative to max eigenvalue

$$\mathbf{B} = \mathbf{P}\,\mathbf{D}\,\mathbf{P}^{-1}$$

D: diagonal matrix of eigenvalues $\{\lambda_1, \ldots \lambda_n\}$

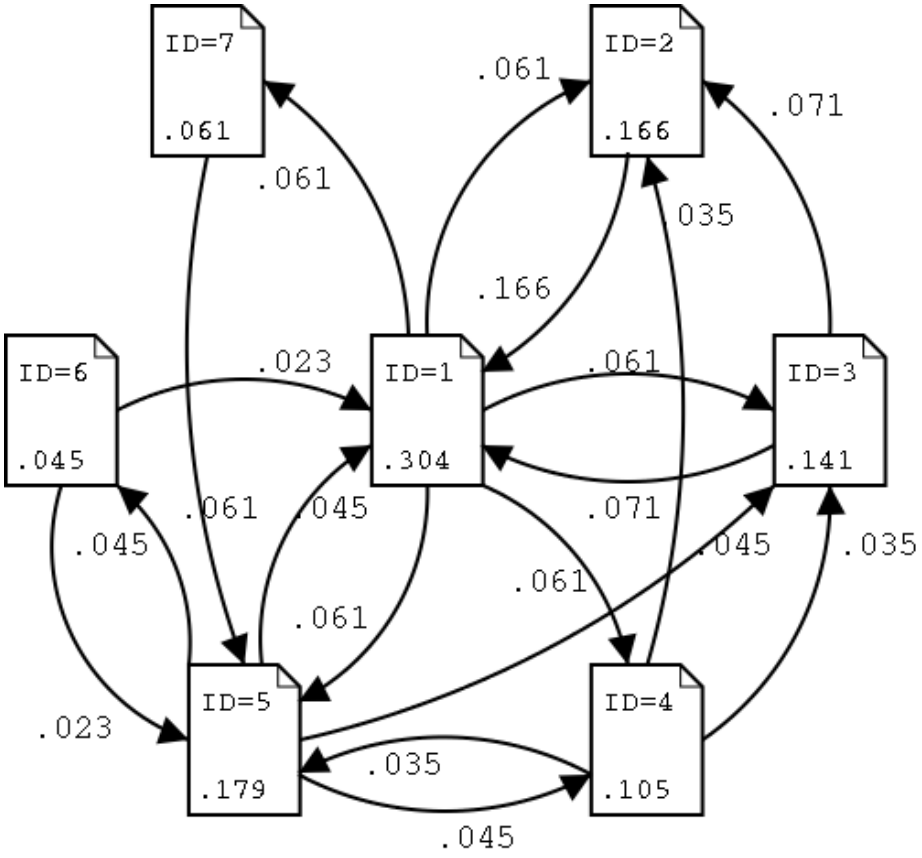
P: regular matrix that consists of eigenvectors

$$\begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \ddots \\ 0 & & \lambda_n \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_n \end{pmatrix}$$

PageRank $\mathbf{r}_1$ = $\begin{pmatrix} 0.69946 \\ 0.38286 \\ 0.32396 \\ 0.24297 \\ 0.41231 \\ 0.10308 \\ 0.13989 \end{pmatrix}$ → normalized $\begin{pmatrix} 0.303514 \\ 0.166134 \\ 0.140575 \\ 0.105431 \\ 0.178914 \\ 0.044728 \\ 0.060703 \end{pmatrix}$

Matrix Notation



PR	ID	OutLink	InLink
0.304	1	2,3,4,5,7	2,3,5,6
0.179	5	1,3,4,6	1,4,6,7
0.166	2	1	1,3,4
0.141	3	1,2	1,4,5
0.105	4	2,3,5	1,5
0.061	7	5	1
0.045	6	1,5	5

- Confirm the result

  # of inlinks from high ranked page

  hard to explain about 5&2, 6&7

- Interesting Topic

  How do you create your homepage highly ranked?

Markov Chain Notation

- Random surfer model
  - Description of a random walk through the Web graph
  - Interpreted as a transition matrix with asymptotic probability that a surfer is currently browsing that page

$$r_t(v) = P(S_t = v) = \sum_w P(S_t = v \mid S_{t-1} = w) P(S_{t-1} = w)$$
$$= \sum_w m_{wv} r_{t-1}(w).$$

$\boldsymbol{r}_t = \mathbf{M}\, \boldsymbol{r}_{t-1}$

M: transition matrix for a first-order Markov chain (stochastic)

Does it converge to a meaningful solution (as $t \to \infty$) regardless of the initial ranks ?

“Rank Sink” Problem

- In general, many Web pages have no inlinks/outlinks
- It results in dangling edges in the graph

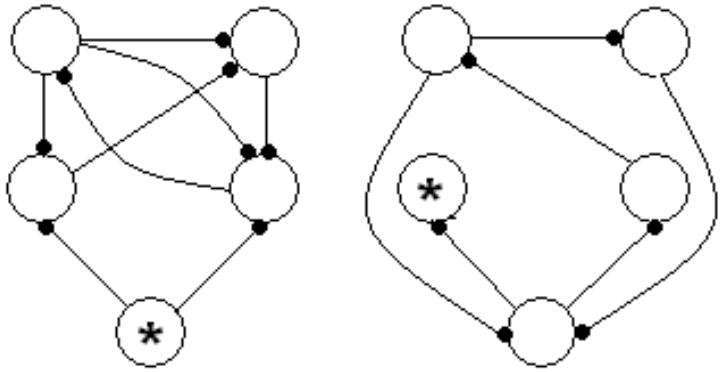
E.g.

no parent → rank 0

M^T converges to a matrix whose last column is all zero

no children → no solution

M^T converges to zero matrix



Modification

- Surfer will restart browsing by picking a new Web page at random

$$\mathbf{M} = (\mathbf{B} + \mathbf{E}), \quad \mathbf{e}_{vw} = \begin{cases} 0 & \text{if } |ch[v]| > 0 \\ \frac{1}{n} & \text{otherwise} \end{cases}$$

E : escape matrix

M : stochastic matrix

- Still problem?
  - It is not guaranteed that **M** is primitive
  - If **M** is stochastic and primitive, PageRank converges to corresponding stationary distribution of **M**

PageRank Algorithm

PAGERANK(M, n, ϵ)

1 $\mathbf{1} \leftarrow [1, \ldots, 1] \in \mathbb{R}^n$
2 $z \leftarrow \frac{1}{n}\mathbf{1}$
3 $x_0 \leftarrow z$
4 $t \leftarrow 0$
5 **repeat**
6 $t \leftarrow t + 1$
7 $x_t \leftarrow M^{\mathrm{T}} x_{t-1}$
8 $d_t \leftarrow \|x_{t-1}\|_1 - \|x_t\|_1$
9 $x_t \leftarrow x_1 + d_t z$
10 $\delta \leftarrow \|x_{t-1} - x_t\|_1$
11 **until** $\delta < \epsilon$
12 **return** x_t

* Page et al, 1998

Pagerank computation methods

- G=(V, E)
- xi:pagerank of page i,
- dj:*out-degree* of page j
- In tabular form: x=Ax where:
- Power method (recursive) converges to the principal eigenvector (probability visiting the page).
- If many pages have no inlinks eigen vector mostly 0's
- Thus

$$x_i = \sum_{(j,i)\in E} d_j^{-1} x_j$$

$$a_{ij} = d_{ij}^{-1} \text{ if } (j,i) \in E$$

$$x = (1-\alpha)e + \alpha A x$$

$$(I - \alpha A)x = (1-\alpha)e$$

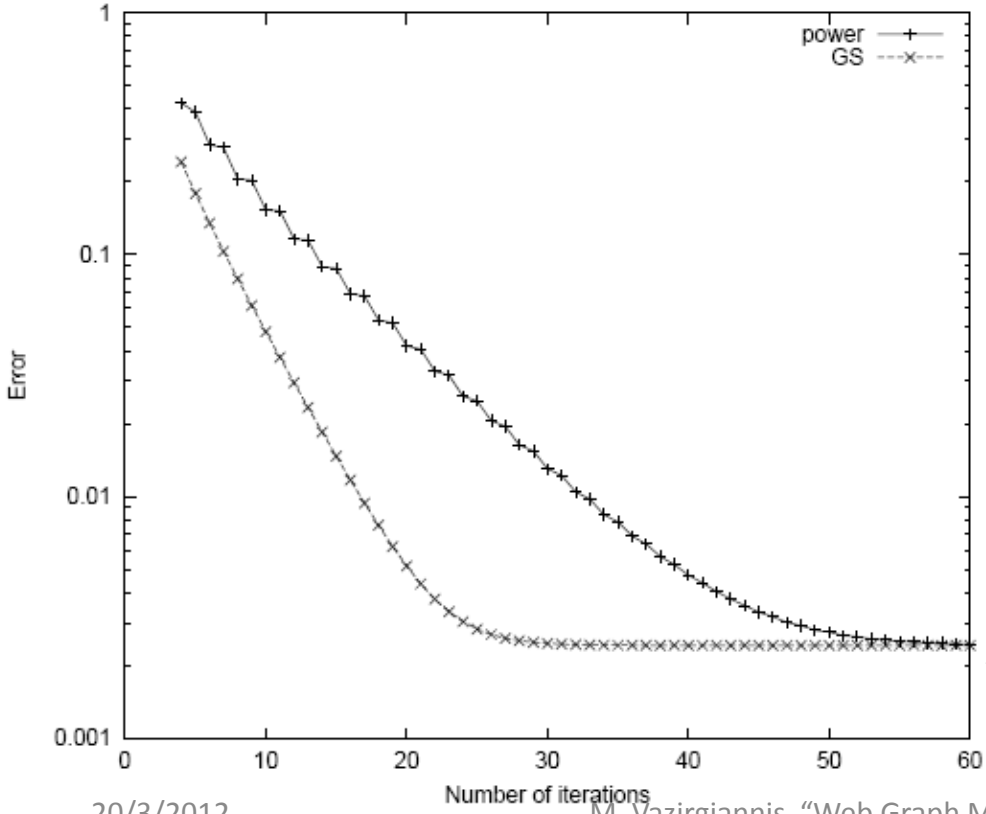
"PageRank Computation and the Structure of the Web: Experiments and Algorithms", Arvind Arasu, Jasmine Novak, Andrew Tomkins & John Tomlin

Pagerank computation methods

- Jacobi iteration
- Gauss-Seidel method

$$x_i^{(k+1)} = (1-\alpha) + \alpha \sum_{(j,i)\in E} a_{ij} x_j^{(k)} \quad \forall i,$$

$$x_i^{(k+1)} = (1-\alpha) + \alpha \sum_{j<i} a_{ij} x_j^{(k+1)} + \alpha \sum_{j>i} a_{ij} x_j^{(k)} \quad \forall i$$



convergence speed

Read:
Amy Nicole Langville and Carl Dean Meyer, Deeper Inside PageRank, Internet Mathematics, 3, 2003

The Largest Matrix Computation in the World

- Computing PageRank can be done via matrix multiplication, where the matrix has several billion rows and columns.
- The matrix is sparse as average number of outlinks is between 7 and 8.
- Setting d = 0.15 or below requires at most 100 iterations to convergence.
- Researchers still trying to speed-up the computation.

Ranking function web search

- Web search engines take into account 100's of features to rank documents assuming a query
- Two important features are
  - The *PageRank* value of the page containing the *query* terms
  - The *relevance* of the term to the specific page
- Given a term *t* the score of a document *d* is computed as:

$$score_t(di) = \mathrm{w}_1(\mathrm{relevance}(\mathrm{t},\mathrm{d}_\mathrm{i})) + \mathrm{w}_2\, pr(\mathrm{d}_\mathrm{i})$$

- Where relevance: tf-idf, BM25 etc.
- In a specific case we used:

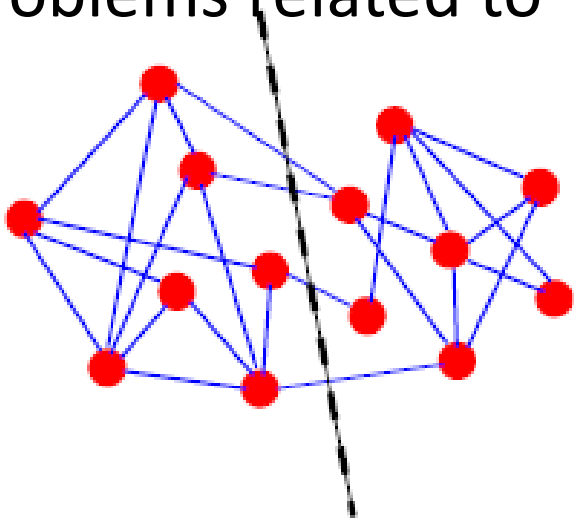
$$score_t(d) = (\mathrm{tf/idf}(\mathrm{t},\mathrm{d}) \cdot \mathrm{title}(\mathrm{t},\mathrm{d}))^{1.5} \cdot pr(d)$$

Tutorial Outline

- Introduction, Motivation
  - Web Data nature (graph, time evolving)
  - Examples of data sets (snap – our data set)
  - Aspects: content, structure and behavior mining.
- Data Preprocessing
  - Content preprocessing
  - Dimensionality reduction algorithms
- Ranking in the context of the Web graph
  - Graph Based ranking
    - Pagerank
    - HITS
  - Pagerank Computation methods
  - Text based ranking
- **Graph clustering for Community detection and evaluation**
  - Graph based Clustering / spectral clustering
  - K-core, D-core structures and related measures
  - Case studies: DBLP, Wikipedia, Epinions

Graph Clusters & Communities

- There is no widely accepted definition
- In general the graph has to be relatively sparse.
- If the graph is too dense then there is no meaning in search of a cluster using the structural properties of the graph.
- Many clustering algorithms or problems related to clustering are NP-hard

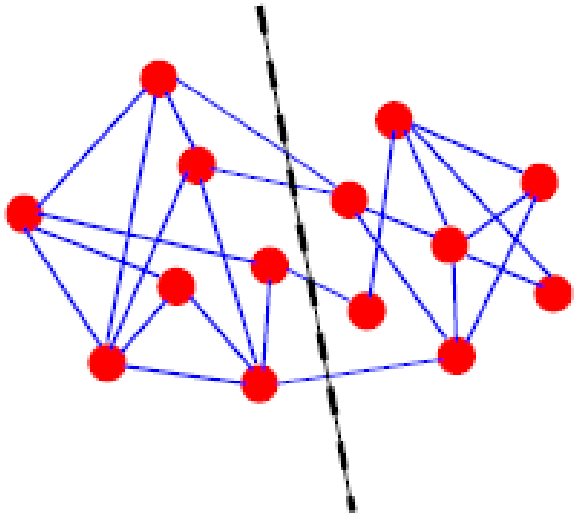


Basic Methodologies

- Graph Partitioning
- Partitional Clustering
- Spectral Clustering
- Divisive Algorithms
- Modularity Based Methodologies

Graph Partitioning

- The general problem is to find k groups of vertices so that the edges between them is minimal.
- k is given as an argument otherwise we end with each vertex as each own cluster.
- Most common algorithms use the Ford and Fulkerson max-flow min-cut theorem



Spectral Clustering

- A similarity matrix -> the adjacency matrix A
- We use the eigenvectors to partition the matrix into clusters.
- Most common methodologies use the Laplacian Matrix L=D-A
  - where D is the diagonal matrix whose element Dii equals the degree of vertex I
- We apply decomposition on L and we use the k eigenvectors with the respective top-k eigenvalues
- If k=2 we use the second eigenvector

Spectral Clustering (k=2)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

$\Lambda =$

0.0
0.4
2.2
2.3
2.5
3.0

$X =$

0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.6
0.4	-0.4	0.9	0.2	-0.4	-0.6
0.4	-0.7	-0.4	-0.8	-0.6	-0.2
0.4	-0.7	-0.2	0.5	0.8	0.9

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

(Here threshold for spliting =0)

x_1	0.2
x_2	0.2
x_3	0.2

x_4	-0.4
x_5	-0.7
x_6	-0.7

Spectral Clustering (Un-normalized)

- Input: Similarity matrix S = Rn×n, number k of clusters to construct.
- Construct a similarity graph by one of the ways define before.
- Let W be its (weighted) adjacency matrix.
- Compute the Laplacian L.
- Compute the top-k eigenvectors $\{u_1, \ldots, u_k\}$ of L.
- Let U = Rn×k be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For i = 1, . . . , n, let $y_1 = R_k$ be the vector corresponding to the i-th row of U.
- Cluster the points (yi) i=1,...,n in R_k, with the k-means algorithm into clusters C1, . . . ,Ck.
- Output: Clusters A1, . . . ,Ak with Ai = {j| yj in Ci}.

Modularity

- Modularity is a metric that can be used to evaluate the quality of a cluster (i.e. its internal density as compared to the average graph density.

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

- Where:
  - m is the number of links in the graph
  - A the adjacency matrix
  - k_i the degree of vertex i
  - c_i the cluster *i* belongs to
  - and δ(ci,cj)=0 if ci==cj, otherwise δ(ci,cj)=1
- The value of the modularity lies in the range [−1,1]
- It is positive if the number of edges within groups exceeds the number expected on the basis of chance.

Community evaluation

- *Community detection* and *evaluation* in graphs is a cornerstone issue .
- Different metrics/ measurements /methods are used
  - Hub/authorities
  - Modularity
  - Density/Diameter/Link distribution etc….
  - Centrality /Betweeness
  - Clustering coefficient
  - Structural cohesion
- On community detection and graph clustering a thorough review is offered by Fortunato [1]

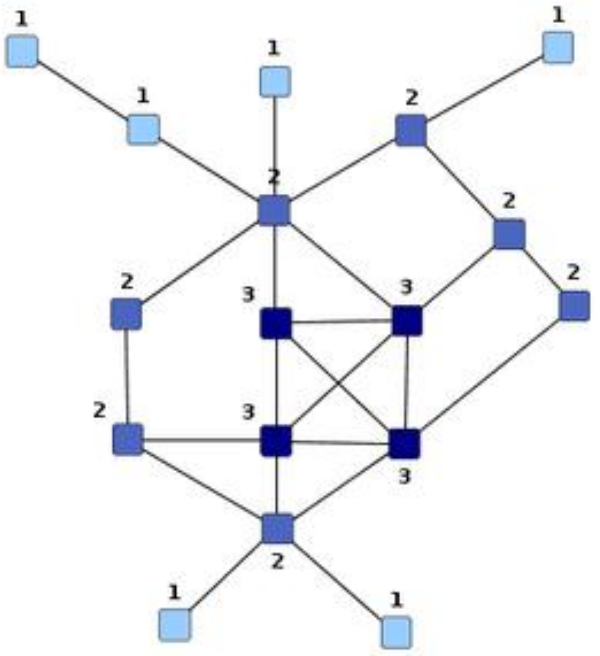
[1]S. Fortunato. Community detection in graphs. Phys. Rep., 486(3-5):75{174, 2010.

Preliminaries - k-Cores

- A *k-core* of a graph G is a maximal sub-graph in which each vertex is adjacent to at least *k* other vertices of the sub-graph
- The maximal k and the size of that (max k)-core can provide a good indication of the cohesiveness of the original graph.

Preliminaries - k-Cores

- A *k-core* of a graph G is a maximal sub-graph in which each vertex is adjacent to at least *k* other vertices of the sub-graph
- The maximal k and the size of that (max k)-core can provide a good indication of the cohesiveness of the original graph.



K-cores

- *G* be a simple undirected graph.
- ***Δ****(G)* the minimum degree of a vertex in *G*.
- Degeneracy of *G* is defined as $\delta^*(G) = \max(\{\Delta(H) | H \subseteq G\}$
- *k*-core : maximum size sub graph *H* of *G* : ***Δ****((H) ≥ k* , k positive integer.
- *Vertex Core number*: maximum *k* for which *v* belongs in the *k-core* of *G*.
- *Subgraph Core index*: for a subset *S* of *G*, it is the maximum *k* for which all vertices of S belong in the *k-core* of *G*.

- The algorithm for computing the *k-cores* follows:
  - ***Procedure*** *$Trim_k(D)$*
  - *Input: An undirected graph D and*
  - *positive integer k*
  - *Output:* ***k-core****(D)*
  - *1. let F D.*
  - *2.* ***while*** *there is a node x in F such that*
    - ***$deg_F(x)<k$***
    - ***delete*** *node x from F.*
  - *3.* ***return*** *F.*

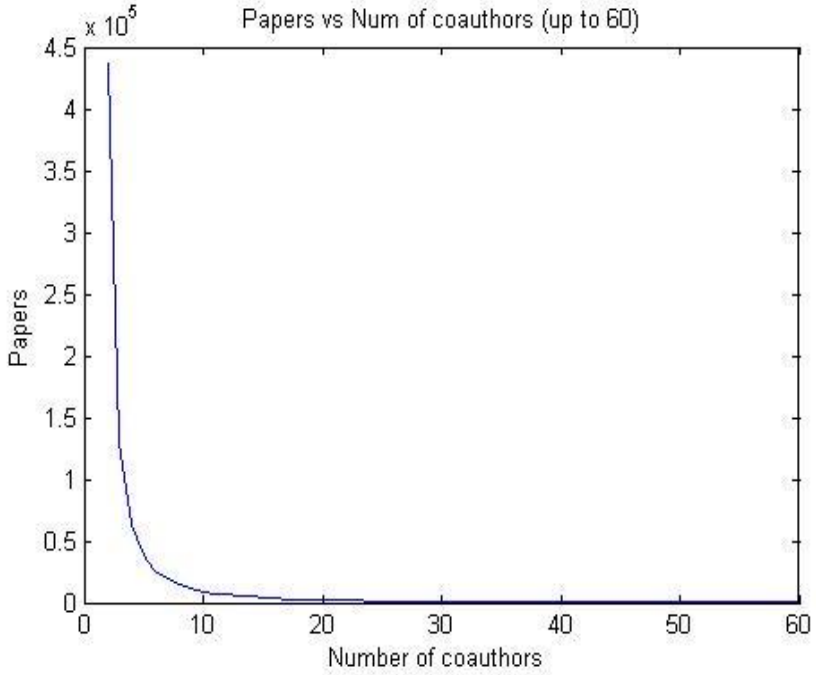
k- Core algorithm complexity

- Our implementation consists of a two steps loop:
  - remove nodes that don’t satisfy the core condition at each time
  - update the remaining nodes
- O(n*k) (n: nodes , k:maximal core number)
- Fast especially in real world data where a sparse graph is usually the case

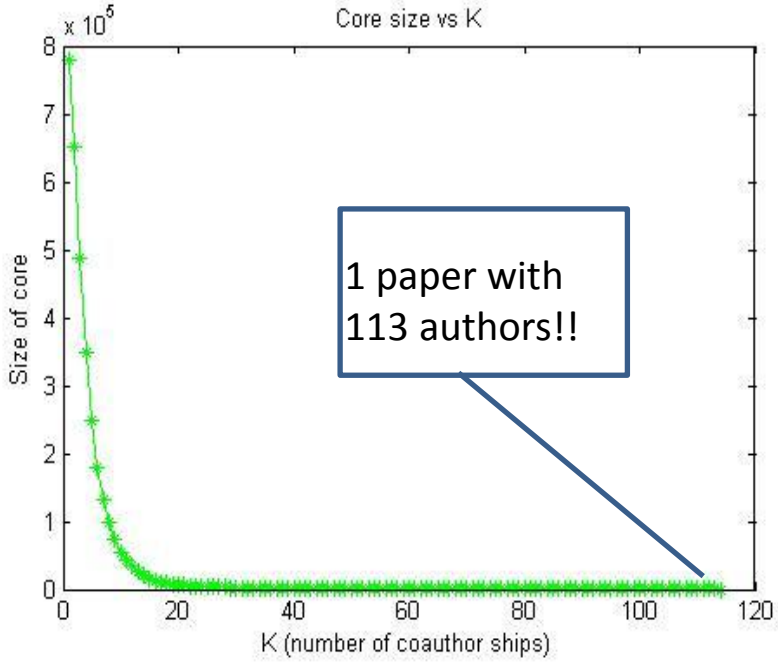
Specific dataset : DBLP

- Taken from the DBLP data set
- Author A and B are connected if the have co-authored at least one paper
- 825 K author-nodes
- About 450k co-authorship links

k-cores for the DBLP coauthorship graph



Distribution of the number of coauthors/paper k-core sizes in
In the unfiltered DBLP coauthorship graph



Distribution of the k-core sizes in the unfiltered DBLP coauthorship graph

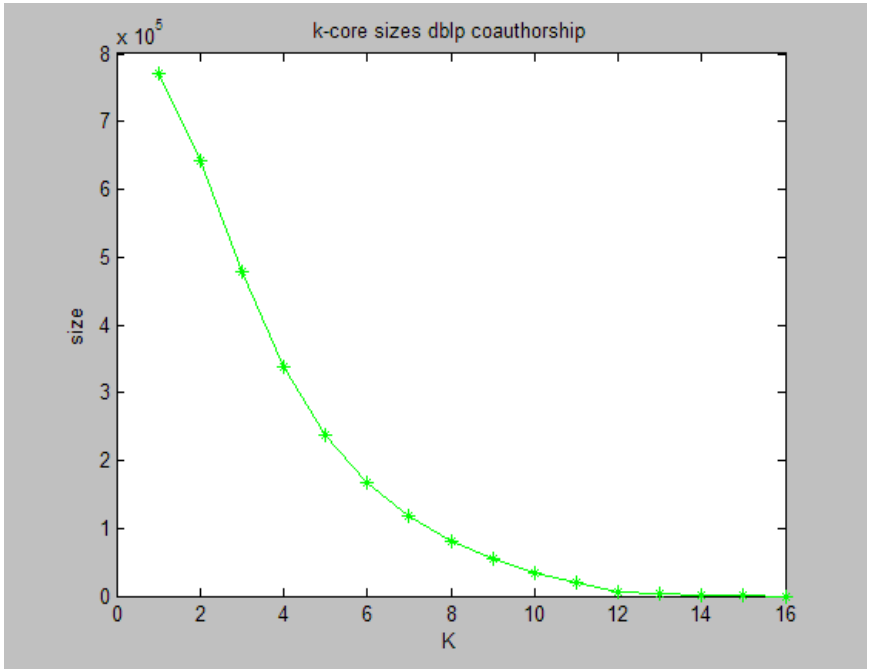
Filtered DBLP K-cores

- Extreme k-core: k=15 (DBLP), 76 authors
- Author ranking metric: max(k)-core that an author belongs to
  - e.g. Paul Erdos : 14
- On the max(k)-core we can identify the "closest" collaborators**: Hop-1 community**
  - Erdos hop-1 :

    Boris Aronov, Daniel J. Kleitman, János Pach, Leonard J. Schulman, Nathan Linial, Béla Bollobás, Miklós Ajtai, Endre Szemerédi, Joel Spencer,Fan R. K. Chung, Ronald L. Graham, David Avis, Noga Alon, László Lovász, Shlomo Moran, Richard Pollack, Michael E. Saks, Shmuel Zaks, Peter Winkler, Prasad Tetali, László Babai

DBLP co-authorship – k-core on filtered graph

- Filtered the papers with more than 15 authors/paper (~1% of the dataset)



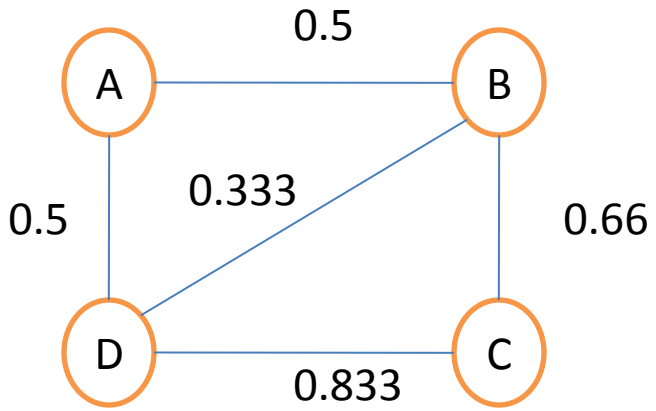
Kurt Mehlhorn	Joseph S. B. Mitchell	Marc J. van Kreveld
Micha Sharir	David Eppstein	Martin L. Demaine
Pankaj K. Agarwal	Erik D. Demaine	Ferran Hurtado
Mark de Berg	Olivier Devillers	Timothy M. Chan
Rolf Klein	Sándor P. Fekete	Oswin Aichholzer
Mark H. Overmars	Henk Meijer	Bettina Speckmann
Herbert Edelsbrunner	Sariel Har-Peled	Jeff Erickson
Stefanie Wuhrer	John Hershberger	Therese C. Biedl
Jack Snoeyink	Alon Efrat	Greg Aloupis
Joseph O'Rourke	Stefan Langerman	David Bremner
Subhash Suri	Bernard Chazelle	Anna Lubiw
Otfried Cheong	Joachim	Esther M. Arkin
Hazel Everett	Gudmundsson	Boris Aronov
Sylvain Lazard	Giuseppe Liotta	Vida Dujmovic
Helmut Alt	Sue Whitesides	Suneeta Ramaswami
Emo Welzl	Christian Knauer	Thomas C. Shermer
Günter Rote	Raimund Seidel	David R. Wood
Leonidas J. Guibas	Michiel H. M. Smid	Perouz Taslakian
Chee-Keng Yap	Tetsuo Asano	John Iacono
Danny Krizanc	David Rappaport	Sergio Cabello
Pat Morin	Vera Sacristan	Sébastien Collette
Jorge Urrutia	Hee-Kap Ahn	Belén Palop
Diane L. Souvaine	Prosenjit Bose	Mirela Damian
Ileana Streinu	Michael A. Soss	Jirí Matousek
Dan Halperin	Godfried T.	Otfried Schwarzkopf
Hervé Brönnimann	Toussaint	Richard Pollack

K-core, issues

- Co-authorship graph: Authors participating in papers with many coauthors get biased credit
- i.e. in the unfiltered case:
  - 1 paper with 113 authors creates the most dense co-authroship collaboration structure
  - for most of the authors was the only paper
- Each author of a paper should get a just credit (i.e. 1/# authors)

Weighting Scheme Example

- Consider Authors A,B,C,D that collaborate in papers as such:
  - 1:A,B
  - 2:B,C,D
  - 3:D,C
  - 4:D,A
- Author D weighted links within each paper:
  - 1:0
  - 2:0.333
  - 3 :0.5
  - 4:0.5



In total D is linked to C with weight:0.833 and to A with weight 0.5

Fractional k-cores (1)

Co-authorship edge weight:

- For every edge *e = {x, x'}* we set
- *The weighted co-authorship affinity among x and x': collaboration !*

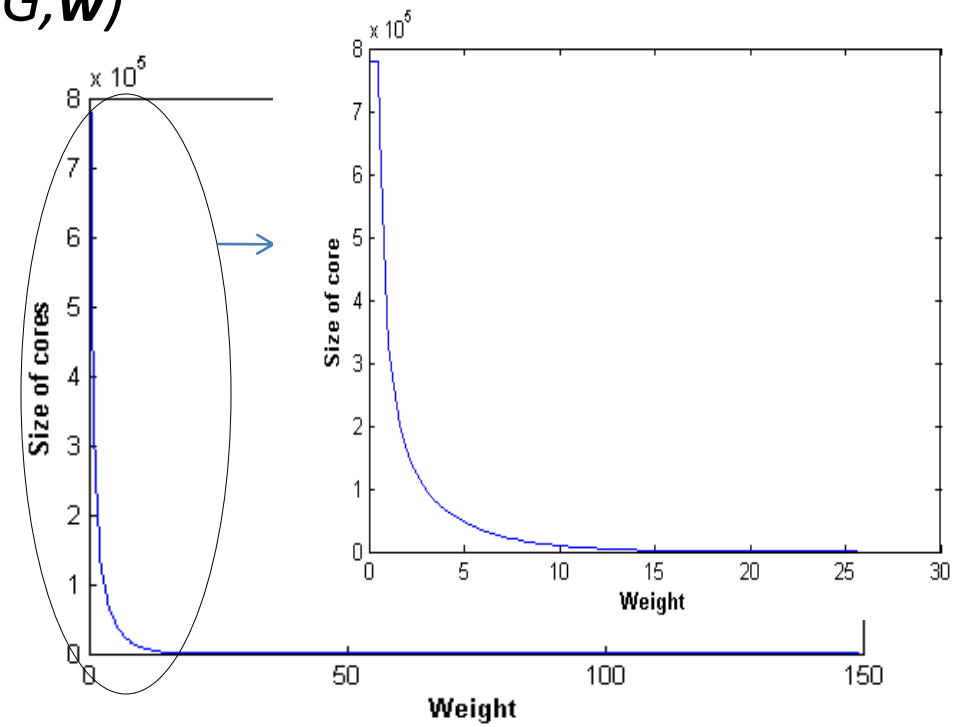
$$w(e) = \sum_{y \in N(x) \cap N(x')} \frac{1}{|N(y)|}$$

Vertex fractional degree. x in *(G,**w**)*

$$\deg_{G,w}(x) = \sum_{e \in E(x)} w(e)$$

- the total co-authorship value of an author

Distribution of the fractional k-core sizes in the DBLP coauthorship graph



Fractional k-cores (2)

- The algorithm does not change -> As fast as before.
- On comparing the ranking:

Name of author	k-core
Serge Abiteboul	14
Christos Faloutsos	14
Gerhard Weikum	14
Christos H. Papadimitriou	14
Paul Erdős	14
Andrew Tanenbaum	12

k –core filtered

Name of author	k-core
Christos H. Papadimitriou	20.8
Serge Abiteboul	20.5
Christos Faloutsos	18.7
Gerhard Weikum	16.3
Paul Erdős	13.9

f-core

FRACTIONAL CORES AND HOP-1 LIST FOR SELECTED AUTHORS.

Author	Max_Core	Size	Hop-1 list
C.H. Papadimitriou	20.80	417	Mihalis Yannakakis 19.62 Erik D. Demaine 0.14 Georg Gottlob 1.0 Moshe Y. Vardi 0.25
G.Weikum	16.30	1506	Hans-Jörg Schek 7.43 Surajit Chaudhuri 5.05 Raghu Ramakrishnan 0.41 Gustavo Alonso 0.43 Divyakant Agrawal 0.29 Yuri Breitbart 1.49 Amr El Abbadi 0.29 Catriel Beeri 0.33 Rakesh Agrawal 0.48 Abraham Silberschatz 0.17 Gautam Das 0.7 S. Sudarshan 0.2 Michael Backes 0.33 Jennifer Widom 0.19 David J. DeWitt 0.19 Stefano Ceri 0.275 Serge Abiteboul 0.33 Umeshwar Dayal 0.17 Michael J. Carey 0.14 …
Tanenbaum	13.0	4016	Maarten van Steen 4.68 Frances M. T. Brazier 0.98 Howard Jay Siegel 0.13 M. Frans Kaashoek 7 Anne-Marie Kermarre 0.25 Robbert van Renesse 5.4 Michael S. Lew 0.02

D-cores

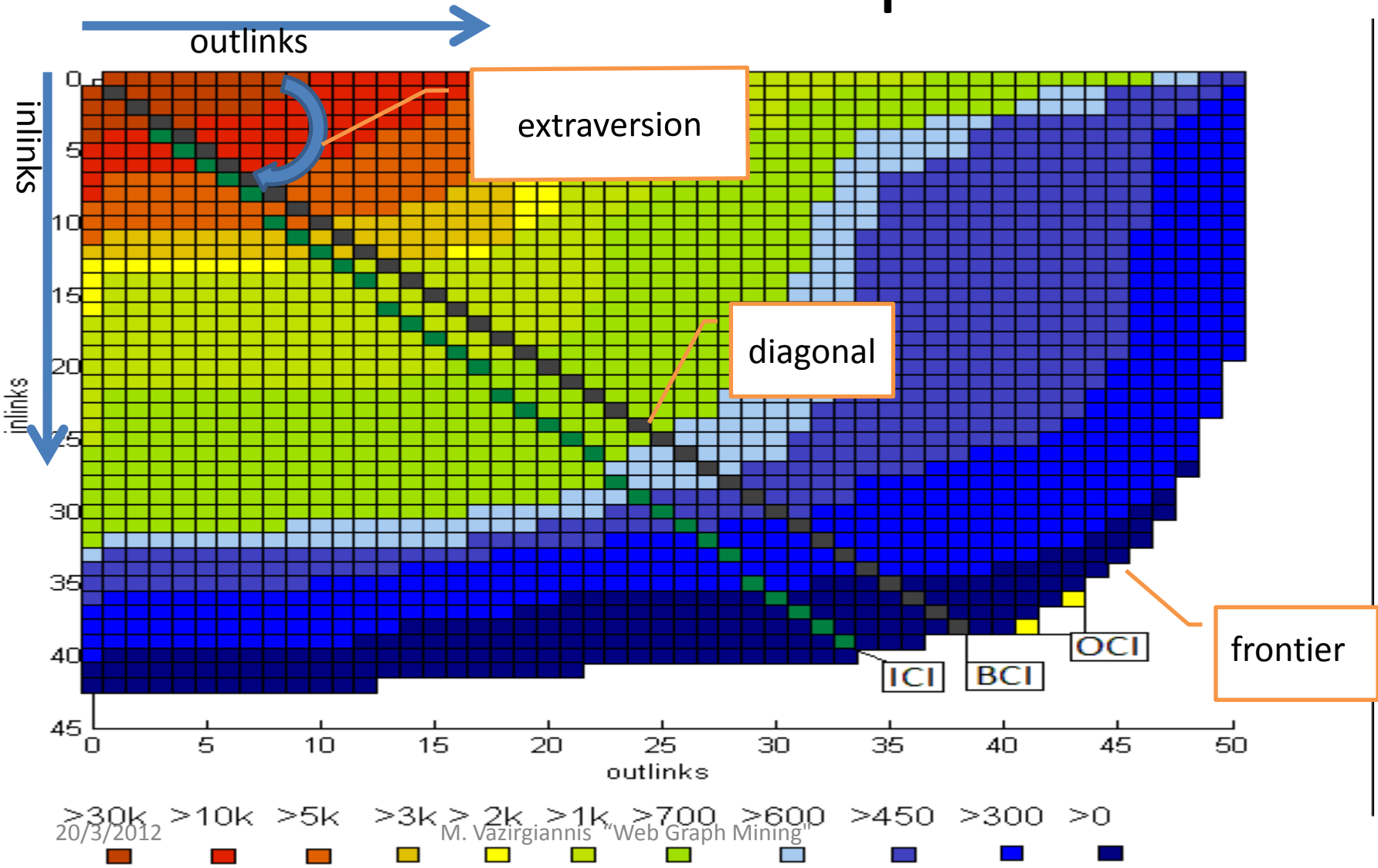
- We extend the k-core concept in directed graphs by applying a limit on in/out edges respectively.
- This provides a two dimensional range to which the cores can “empty-up”.
- The additional information can give us a more specific view of the cohesiveness and the “social” behavior of the graph

D-Core Matrix & relevant indices

- DC(k,l) (D) is the (k,l) D-core of the graph G
  - for each $_{k,l}$: $\mathbf{dc_{k,l}} = |DC(k,l)|$
- D-core matrix: (infinite) $D(k,l)=(dc_{k,l})$, k,l integers
- **Frontier:** $F(D) = \{(k;l): dc_{k,l} > 0 \ \& \ dc_{k+1,l+1} = 0 \}$
- Balanced collaboration index(**BCI**) :
  - Intersection of diagonal with frontier
- Optimal collaboration index (**OCI**) :
  - DC(k,l) where max((k+l)/2)
- Inherent collaboration index (**ICI**):
  - All cores on the angle defined by the average inlinks/outlinks ratio
- Average collaboration index (**ACI**):
  - The average angle of all the D-cores on the frontier.

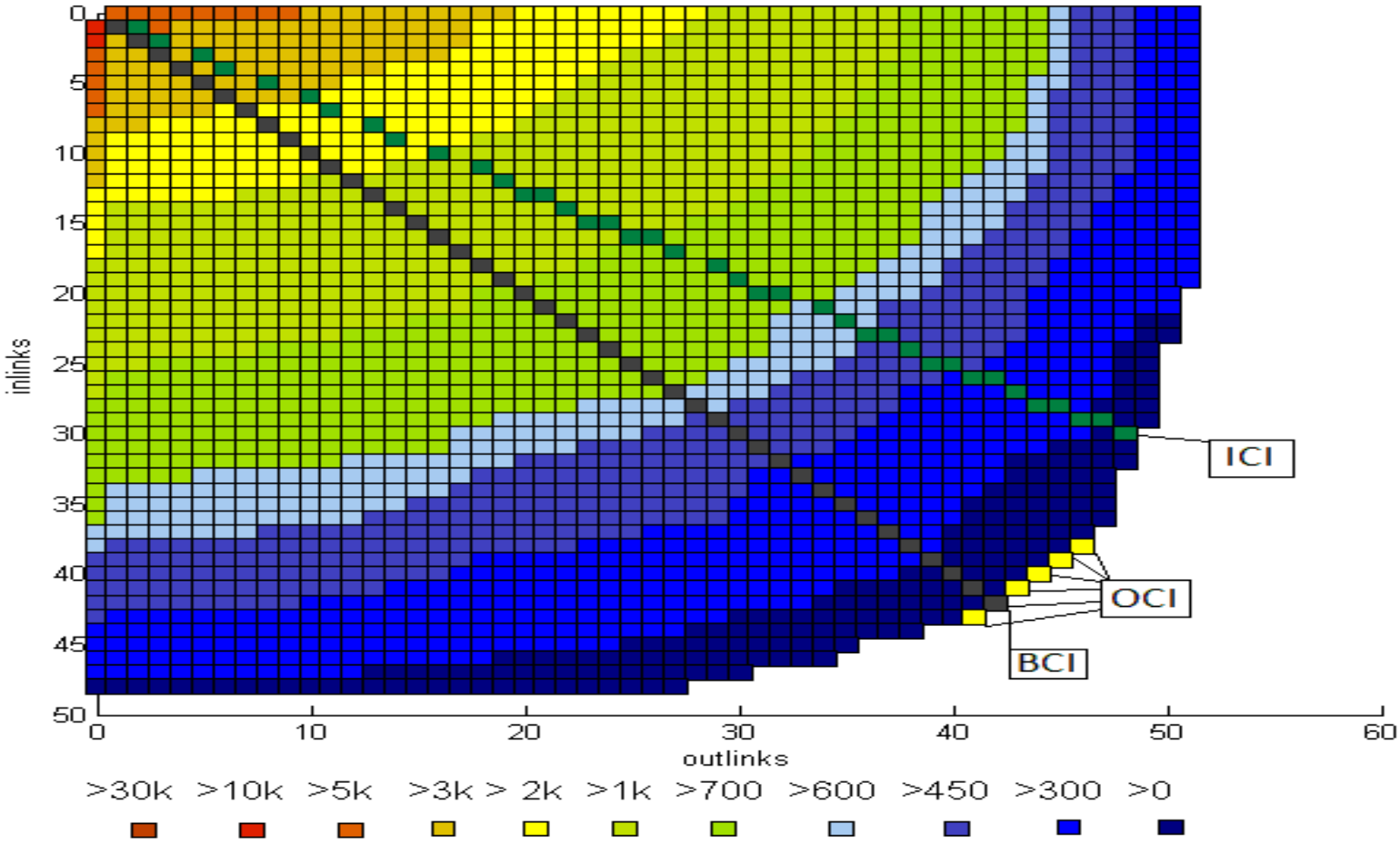
$$\mathrm{ACI}(D) = \frac{\sum_{(k,l)\in F(D)}(k+l)}{2\cdot|F(D)|}.$$

D-core matrix Wikipedia



The extreme Dcore(38,41) contains 237 pages

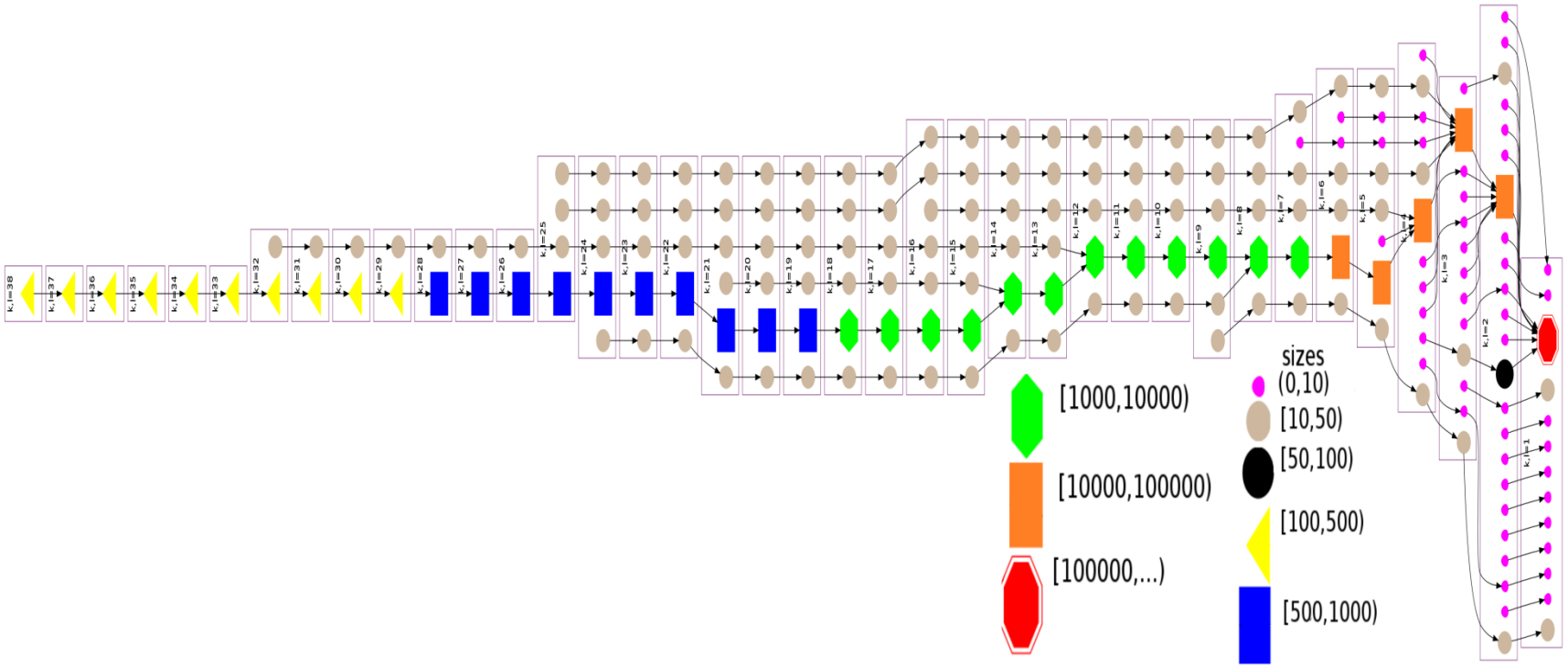
D-core matrix for DBLP



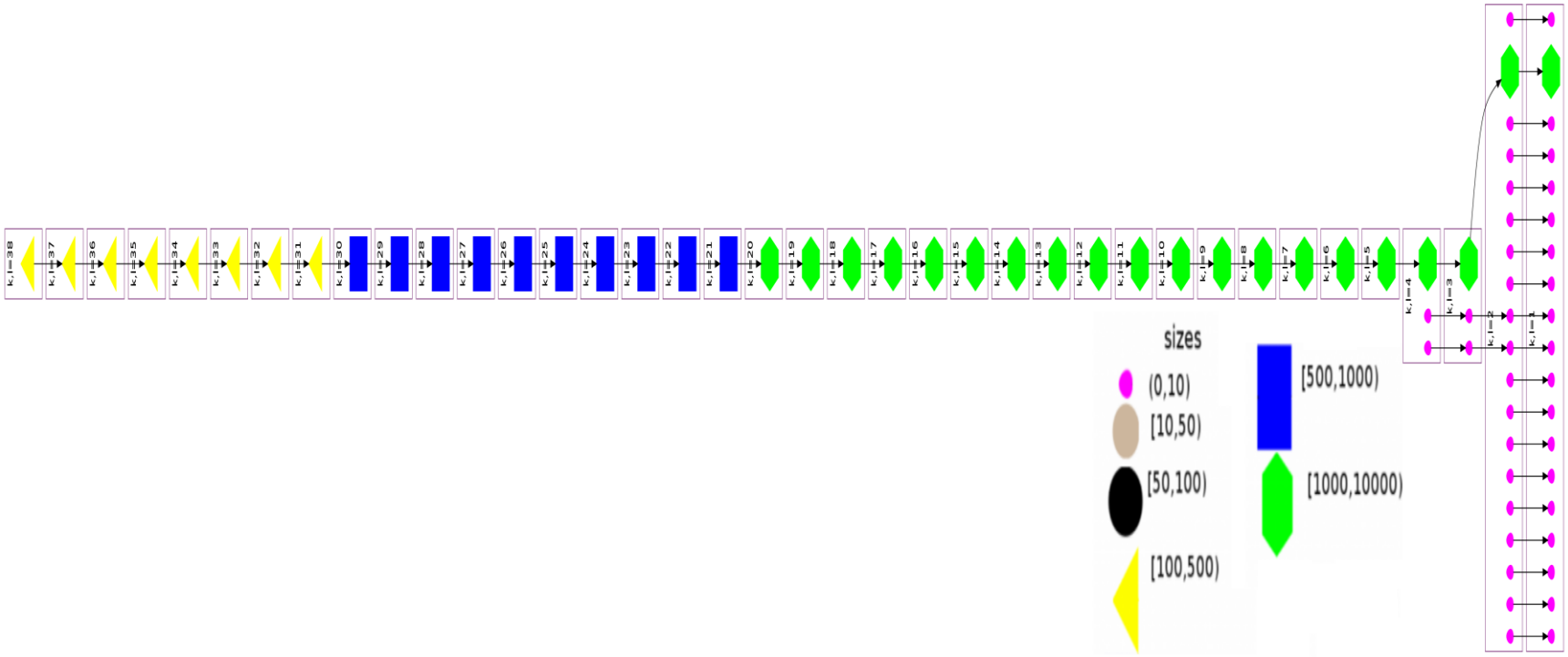
Strongly Connected Components (SCC)

- The SCC's are not usually used for community detection but in this case serve well for indicating how the communities would survive throughout the cores.
- Following we can see the SCC's one each data set thought out the D-Cores on the "diagonal" direction (where the limit of in/out edges is equal)
- Notice: Despite the difference in "nature" and size in the graphs we can see a similar behavior regarding the "length" of the diagonal (DC(k,k))

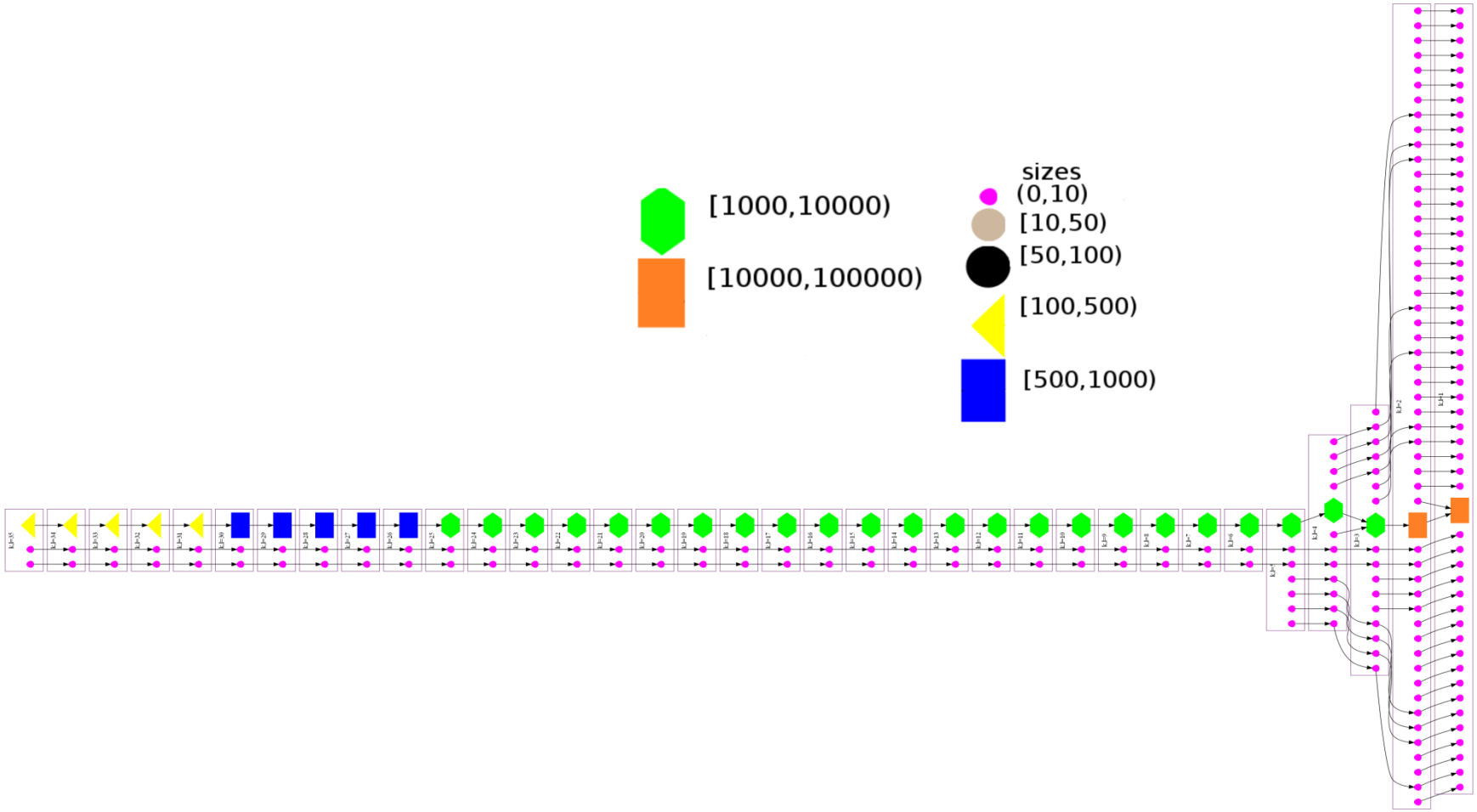
Wikipedia SCCs



DBLP SCCs



Epinions SCCs



D-cores - features

- As we move “down” to more dense cores only the most cohesive parts of the graph survive
- Thus it is safe to assume that the remaining communities will contain the most “basic” nodes of each community
- The same notion should be applicable in both directed and undirected graphs

Conclusions

- Graph Mining is fertile research area with prominent industrial applications:
  - Web Search
  - Social networks & community mining
- Contributions
  - a novel metric for evaluating the cohesiveness of communities based on the k-core structure
  - an innovative extension of the k-core concept assigning weights on the edges and
  - an extended experimental evaluation in the case of the DBLP
- Promising directions – Further work
  - Extending the k-core concept for signed graphs
  - Further exploration of communities

Online demo available

(k-cores, f-cores on DBLP)

http://www.graphdegeneracy.org/

Thank you!

For more information check:

http://www.lix.polytechnique.fr/~mvazirg

Graph Clustering - References

[1] http://en.wikipedia.org/wiki/Jaccard_index

[2] ROCK: A Robust Clustering Algorithm for Categorical Attributes Sudipto Guha, Rajeev Rastogi, Kyuseok Shim

[3] http://en.wikipedia.org/wiki/Gomory%E2%80%93Hu_tree

[4] http://en.wikipedia.org/wiki/Minimum_cut

[5] http://en.wikipedia.org/wiki/Cut_%28graph_theory%29

[6] http://en.wikipedia.org/wiki/Edmonds%E2%80%93Karp_algorithm

[7] http://mathworld.wolfram.com/LaplacianMatrix.html

[8] A Tutorial on Spectral Clustering , Ulrike von Luxburg

[9] Community structure in social and biological networks M. Girvan and M. E. J. Newman

[10] Community detection in graphs, Santo Fortunato , Physics Reports Volume 486, Issues 3-5, February 2010, Pages 75-174

[11] Modularity from Fluctuations in Random Graphs and Complex Networks Roger Guimera, Marta Sales-Pardo, and Luıs A. Nunes Amara

References - Pagerank prediction & Web Dynamics

1. Baeza-Yates, R. A., Castillo, C., & Saint-Jean, F. (2004). Web Dynamics, Structure, and Page Quality. In M. Levene, & A. Poulovasillis (Eds.), *Web Dynamics* (pp. 93-112). Springer.
2. Broder, A. Z., Lempel, R., Maghoul, F., & Pedersen, J. (2006). Efficient PageRank approximation via graph aggregation. *Information Retrieval , 9* (2), 123-138.
3. Chen, Y.-Y., Gan, Q., & Suel, T. (2004). Local Methods for Estimating PageRank Values. *Proceedings CIKM.* Washington, USA.
4. Chien, S., Dwork, C., Kumar, R., Simon, D. R., & Sivakumar, D. (2003). Link Evolution: Analysis and Algorithms. *Internet Mathematics , 1* (3), 277-304.
5. Davis, J. V., & Dhillon, I. S. (2006). Estimating the global PageRank of Web communities. *Proceedings KDD.* Philadelphia, USA.
6. Deshpande, M., & Karypis, G. (2004). Selective Markov models for predicting Web page accesses. *ACM Trans. Internet Technol.* , 163-184.
7. Langville, A. N., & Meyer, C. D. (2004). Updating PageRank with iterative aggregation. *Proceedings of the 13th international World Wide Web conference on Alternate track papers \& posters* (pp. 392-393). New York, USA: ACM.
8. Piatetsky-Shapiro, G., & Connell, C. (1984). Accurate estimation of the number of tuples satisfying a condition. *SIGMOD Rec. , 14* (2), 256-276.
9. Sayyadi, H., & Getoor, L. (2009). Future Rank: Ranking Scientific Articles by Predicting their Future PageRank. *2009 SIAM International Conference on Data Mining (SDM09)* (pp. 533-544). Sparks, Nevada: SIAM.
10. Vazirgiannis, M., Drosos, D., Senellart, P., & Vlachou, A. (2008). Web Page Rank Prediction with Markov Models. *WWW poster.* Beijing, China.
11. Vlachou, A., Berberich, K., & Vazirgiannis, M. (2006). Representing and quantifying rank-change for the Web graph. *Algorithms and Models for the Web-Graph, Fourth International Workshop, WAW 2006. 4936*, pp. 157-165. Banff, Canada: Springer.
12. Zacharouli, P., Titsias, M., & Vazirgiannis, M. (2009). Web Page Rank Prediction with PCA and EM Clustering. *Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph* (pp. 104-115). Barcelona, Spain: Springer-Verlag Berlin, Heidelberg.

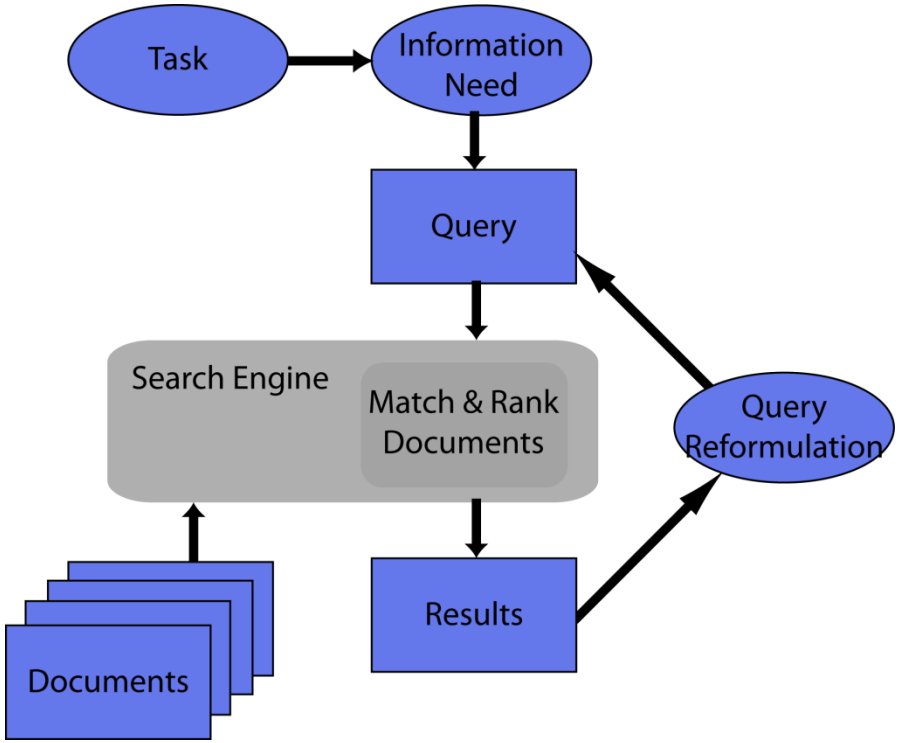
Backup slides

Text based ranking - Overview

- (Typical) Search Process
- Boolean Retrieval Model
- Vector Space Model
- BM25 Ranking Function
- Bayesian and Combinatorial Models
- Ranking with Document fields
- Ranking with Proximity
- Learning to Rank

(Typical) Search process

- A User performs a ***task*** and has an ***information need***, from which a ***query*** is formulated.
- A ***search engine*** typically processes textual queries and retrieves a (ranked) list of ***documents***
- The user obtains the ***results*** and may ***reformulate*** the query



Boolean Retrieval Model

- Documents are represented as sets of terms
- Queries are Boolean expressions with precise semantics
- Retrieval is precise and the answer is a set of documents
  - Documents are *not ranked*
  - Simplest retrieval model
- Example:

  D1 = {web, text, mining}

  D2 = {data, mining, text, clustering}

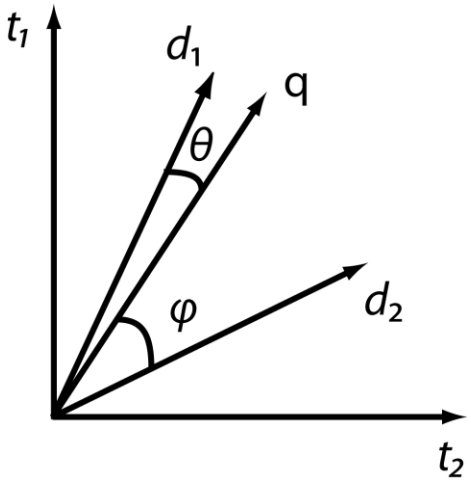
  D3 = {frequent, itemset, mining, web}
  - For query q = *web ∧ mining* answer is { D1, D3 }
  - For query *q = text ∧ (clustering ∨ web)* answer is { D1, D2 }
- Extended Boolean Retrieval Model **(Salton et al., 1983)**
  - Ranks documents according to similarity to query
  - Matches documents that satisfy the Boolean query partially

Vector Space Model

- Document ***d*** and query ***q*** are represented as k-dimensional vectors $d = (w_{1,d}, \ldots, w_{k,d})$ and $q = (w_{1,q}, \ldots, w_{k,q})$
  - Each dimension corresponds to a term from the collection vocabulary
  - Independence between terms
  - $w_{i,q}$ is the weight of i-th vocabulary word in ***q***
  - **(Salton & McGill, 1983; Baeza-Yates & Ribeiro-Neto, 1999)**

- Degree of similarity between ***d*** and ***q*** is the cosine of the angle between the two vectors

$$sim(d,q) = \frac{\vec{d} \bullet \vec{q}}{|\vec{d}| \times |\vec{q}|} = \frac{\sum_{t=1}^{k} w_{t,d} \times w_{t,q}}{\sqrt{\sum_{t=1}^{k} w_{t,d}^2} \times \sqrt{\sum_{t=1}^{k} w_{t,q}^2}}$$



- Generalized Vector Space Model **(Wong et al., 1985)**
  - Incorporates dependencies between terms
  - Using semantics **(Tsatsaronis & Panagiotopoulou, 2009)**

Term weighting in VSM

- Term weighting with tf-idf (and variations)

$$w_{t,d} = tf_{t,d} \times idf_t$$

- tf models the importance of a term in a document

$$tf_{t,d} = f_{t,d} \qquad tf_{t,d} = \frac{f_{t,d}}{\max(f_{s,d})}$$

  - $f_{t,d}$ is the frequency of term *t* in document *d*

- idf models the importance of a term in the document collection
  - Logarithm base not important
  - Theoretically derived from Probabilistic IR, Information Theory **(Robertson, 2004; Papineni, 2001; Metzler, 2008)**

$$idf_t = \log \frac{N}{n_t} \qquad idf_t = \log \frac{N - n_t + 0.5}{n_t + 0.5}$$

  - *N* is the total number of documents, n_t is the document frequency of term *t*

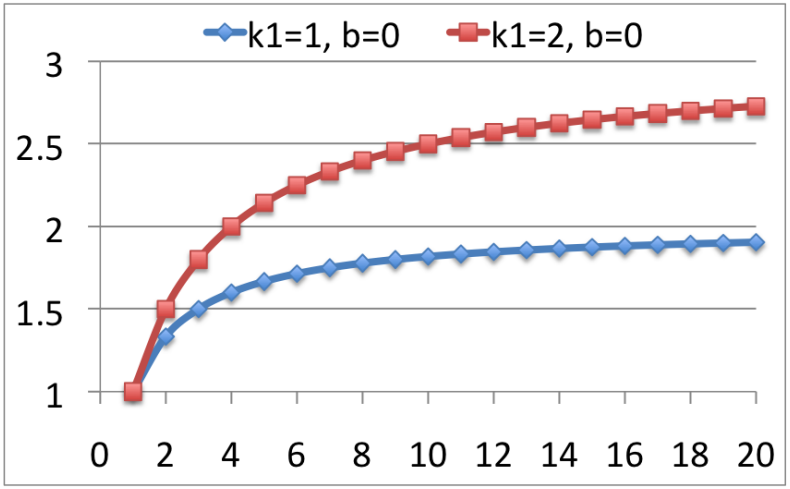
BM25 Ranking Function

- tf-idf like ranking function assuming bag-of-words document representation **(Robertson et al., 1994)**

$$score(d,q) = \sum_{t \in d \cap q} idf_t \times \frac{tf_{t,d} \cdot (k_1 + 1)}{tf_{t,d} + k_1 \cdot \left(1 - b + b \frac{len_d}{avglen}\right)} \quad \text{where} \quad idf_t = \log \frac{N - n_t + 0.5}{n_t + 0.5}$$

  - len_d is the length of document d
  - *avglen* is the average document length in the collection

- Values of parameters k_1 and b depend on collection/task
  - k_1 controls term frequency saturation
  - b controls length normalization
  - Default values: $k_1 = 1.2$ and $b = 0.75$



Bayesian & Combinatorial Models

- Language Models for IR based on a Bayesian approach **(Ponte & Croft, 1998; Hiemstra, 1998; Laferty & Zhai, 2001)**
  - Generate a language model M_d for every document d
  - Rank documents according to probability $P(q|M_d)$ that they generate query q
  - Principled integration of the importance of documents (e.g. PageRank), as prior probability $P(d)$
- Divergence From Randomness based on a combinatorial approach **(Amati, 2003)**
  - Information theoretic framework for generating ranking functions

$$w_{d,q} = \sum_{t \in q \cap d} (-\log P_1) \cdot (1 - P_2)$$

  - Models consist of 3 components:
  - **Randomness model** computes probability P_1 that t appears with given frequency in d
  - **Risk/Information Gain Model** computes probability P_2 that t with given frequency in d will occur one time more
  - **Term Frequency Normalisation** adjusts term frequency with respect to document length

Ranking with Document fields

- Documents often have content in different fields
  - Metadata in PDF or MS Office files
  - Title, headings, emphasized text, anchor text of incoming hyperlinks in Web pages
- Rank by
  - Scoring each field separately and combining the scores
    - Difficult to interpret: combining non-linear transformations of term frequencies
  - Computing weighted sum of field term-frequencies, then compute score
    - Per-field term frequency normalization: e.g. no need to discount anchor text frequencies, because each anchor is a vote for the Web page
- Extension of BM25 **(Zaragoza et al. 2004)** and DFR models **(Macdonald et al., 2005)** with per-field normalization

Ranking with Proximity

- Query terms that appear near each other contribute more to the document's score
  - Requires position information in the search engine index
  - Score both individual query terms and pairs of query terms

$$score(d,q) = \sum_{t \in q} score(d,t) + \sum_{t_i, t_j \in q, t_i \neq t_j} score(d, t_i, t_j)$$

- Count occurrence of pair of terms if they occur within window of *w* terms

- Random Markov Fields to model dependencies **(Metzler & Croft, 2005)**
  - Sequential Dependence: only consecutive terms
  - Full Dependence: all possible pairs of terms
- Extensions to BM25 and DFR **(Büttcher & Clarke, 2006; Peng et al., 2007)**

- Using global counts for pairs of terms expensive to compute and store
  - Not always effective **(Macdonald & Ounis, 2010)**

Learning To Rank

- Web search engines use hundreds of features to rank Web pages
  - Text-based features (e.g. BM25 scores), term proximity, document structure,
  - Link-based features (e.g. PageRank),
  - spam metrics, freshness, click-through rates
- Limitations of typical IR models
  - Parameter tuning can be difficult
  - Cannot combine large number of features to obtain more effective ranking
- Machine Learning from
  - Points: relevance judgment for query-document pair **(Crammer & Singer, 2001)**
    - Doc A is relevant, Doc B is not relevant, Doc C is somehow relevant
  - Pairs: preferences between pairs of documents **(Burges et al., 2005)**
    - Doc A is better than Doc B
  - Lists: maximize average evaluation measure for many queries **(Yue et al., 2007)**
    - Hard problem, evaluation measures are not always continuous, approximations are required