Applications
of
Nonnegative Matrices:
Ranking
and
Clustering

Amy Langville
Mathematics Department
College of Charleston
Collaborators

- Carl Meyer, N. C. State University
- David Gleich, Stanford University
- Michael Berry, University of Tennessee-Knoxville
- College of Charleston (current and former students)
  - Luke Ingram
  - Ibai Basabe
  - Kathryn Pedings
  - Neil Goodson
  - Colin Stephenson
  - Emmie Douglas
Nonnegative Matrices

- Ranking webpages
- Ranking sports teams
- Recommendation systems
- Meta-algorithms
  - Rank and rating aggregation
  - Cluster aggregation
Ranking webpages
Google's PageRank and Beyond
THE SCIENCE OF SEARCH ENGINE RANKINGS

AMY N. LANGVILLE and CARL D. MEYER
1998: enter Link Analysis

- uses hyperlink structure to focus the relevant set
- combine traditional IR score with popularity score
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

• It’s huge.
  — over 10 billion pages, average page size of 500KB
  — 20 times size of Library of Congress print collection
  — Deep Web - 400 X bigger than Surface Web
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

• It’s huge.
  – over 10 billion pages, average page size of 500KB
  – 20 times size of Library of Congress print collection
  – Deep Web - 400 X bigger than Surface Web

• It’s dynamic.
  – content changes: 40% of pages change in a week, 23% of .com change daily
  – size changes: billions of pages added each year
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It’s huge.
  - over 10 billion pages, average page size of 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 400 X bigger than Surface Web

- It’s dynamic.
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year

- It’s self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It’s huge.
  - over 10 billion pages, average page size of 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 400 X bigger than Surface Web

- It’s dynamic.
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year

- It’s self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!

A Herculean Task!
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It’s huge.
  - over 10 billion pages, each about 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 400 X bigger than Surface Web

- It’s dynamic.
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year

- It’s self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!

- Ah, but it’s hyperlinked!
  - Vannevar Bush’s 1945 memex
Elements of a Web Search Engine

WWW

Crawler Module

Page Repository

User

Indexing Module

Indexes

Ranking Module

Queries

Results

Query Module

Special-purpose indexes

Content Index

Structure Index

query-independent
The Ranking Module (generates popularity scores)

- Measure the importance of each page
The Ranking Module (generates popularity scores)

- Measure the importance of each page
- The measure should be Independent of any query
  - Primarily determined by the link structure of the Web
  - Tempered by some content considerations
The Ranking Module (generates popularity scores)

- Measure the importance of each page
- The measure should be Independent of any query
  - Primarily determined by the link structure of the Web
  - Tempered by some content considerations
- Compute these measures off-line long before any queries are processed
The **Ranking Module** (generates popularity scores)

- Measure the importance of each page
- The measure should be Independent of any query
  - Primarily determined by the link structure of the Web
  - Tempered by some content considerations
- Compute these measures off-line long before any queries are processed
- Google’s PageRank© technology distinguishes it from all competitors
The Ranking Module (generates popularity scores)

- Measure the importance of each page

- The measure should be Independent of any query
  - Primarily determined by the link structure of the Web
  - Tempered by some content considerations

- Compute these measures off-line long before any queries are processed

- Google’s PageRank© technology distinguishes it from all competitors

Google’s PageRank = Google’s $$$$$
Take Your Pick

Amount of Internet search results that Web surfers typically scan before selecting one

A few search results* 23%
First page of search results 39%
First two pages 19%
More than first three pages 10%
First three pages 9%

*Top results without reading through the whole page

Note: Sample size is 2,369 people
Sources: JupiterResearch; iProspect
Business intelligence - Wikipedia, the free encyclopedia
Business intelligence (BI) is a business management term which refers to applications and technologies which are used to gather, provide access to, ... en.wikipedia.org/wiki/Business_intelligence - 43k - Cached - Similar pages

Business Intelligence - The Resource for Business Intelligence
The Business Intelligence resource for business and technical professionals covering a wide range of topics including Performance Management, Data Warehouse ... www.businessintelligence.com/ - 74k - Apr 15, 2007 - Cached - Similar pages

Business Intelligence and Performance Management Software ...
Business intelligence and business performance management software. Reporting, analytics software, budgeting software, balanced scorecard software, ... Stock quote for COGN
www.cognos.com/ - 32k - Cached - Similar pages

Oracle Business Intelligence Solutions
The First Comprehensive, Cost-Effective BI Solution Only Oracle delivers a complete, pre-integrated technology foundation to reduce the cost and complexity ... www.oracle.com/solutions/ business_intelligence/index.html - 55k - Cached - Similar pages

Business Intelligence - Management Best Practice Reports
Business Intelligence: Providers of independent reports containing best practice advice, proprietary research findings and case studies for senior managers ...
www.business-intelligence.co.uk/ - 18k - Cached - Similar pages

Intelligent Enterprise: Better Insight for Business Decisions,

The Next Frontiers

The New Age of

Google

The Search Giant Has Changed Our Lives. Can Anybody Catch These Guys? By Steven Levy

PLUS: The Future of Digital Voting

Google founders Larry Page and Sergey Brin
Google’s PageRank
(Lawrence Page & Sergey Brin 1998)

The Google Goals

• Create a PageRank $r(P)$ that is not query dependent
  ▶ Off-line calculations — No query time computation

• Let the Web vote with in-links
  ▶ But not by simple link counts
    — One link to $P$ from Yahoo! is important
    — Many links to $P$ from me is not

• Share The Vote
  ▶ Yahoo! casts many “votes”
    — value of vote from Yahoo! is diluted
  ▶ If Yahoo! “votes” for $n$ pages
    — Then $P$ receives only $r(Y)/n$ credit from $Y$
Google’s PageRank

(Lawrence Page & Sergey Brin 1998)

The Google Goals

- Create a PageRank $r(P)$ that is not query dependent
  - Off-line calculations — No query time computation

- Let the Web vote with in-links
  - But not by simple link counts
    - One link to $P$ from Yahoo! is important
    - Many links to $P$ from me is not

- Share The Vote
  - Yahoo! casts many “votes”
    - value of vote from Yahoo! is diluted
  - If Yahoo! “votes” for $n$ pages
    - Then $P$ receives only $r(Y)/n$ credit from $Y$
Google’s PageRank

(Lawrence Page & Sergey Brin 1998)

The Google Goals

- Create a PageRank \( r(P) \) that is not query dependent
  - Off-line calculations — No query time computation
- Let the Web vote with in-links
  - But not by simple link counts
    - One link to \( P \) from Yahoo! is important
    - Many links to \( P \) from me is not
- Share The Vote
  - Yahoo! casts many “votes”
    - value of vote from Yahoo! is diluted
  - If Yahoo! “votes” for \( n \) pages
    - Then \( P \) receives only \( r(Y)/n \) credit from \( Y \)
Google’s PageRank

(Lawrence Page & Sergey Brin 1998)

The Google Goals

• Create a PageRank $r(P)$ that is not query dependent
  ▶ Off-line calculations — No query time computation

• Let the Web vote with in-links
  ▶ But not by simple link counts
    — One link to $P$ from Yahoo! is important
    — Many links to $P$ from me is not

• Share The Vote
  ▶ Yahoo! casts many “votes”
    — value of vote from Yahoo! is diluted
  ▶ If Yahoo! “votes” for $n$ pages
    — Then $P$ receives only $r(Y)/n$ credit from $Y$
The Definition

\[ r(P) = \sum_{P \in B_P} \frac{r(P)}{|P|} \]

\( B_P = \{ \text{all pages pointing to } P \} \)

\( |P| = \text{number of out links from } P \)
PageRank

The Definition

\[ r(P) = \sum_{P \in B_P} \frac{r(P)}{|P|} \]

\[ B_P = \{ \text{all pages pointing to } P \} \]

\[ |P| = \text{number of out links from } P \]

Successive Refinement

Start with \( r_0(P_i) = \frac{1}{n} \) for all pages \( P_1, P_2, \ldots, P_n \)
PageRank

The Definition

\[ r(P) = \sum_{P \in B_P} \frac{r(P)}{|P|} \]

\[ B_P = \{ \text{all pages pointing to } P \} \]

\[ |P| = \text{number of out links from } P \]

Successive Refinement

Start with \( r_0(P_i) = \frac{1}{n} \) for all pages \( P_1, P_2, \ldots, P_n \).

Iteratively refine rankings for each page

\[ r_1(P_i) = \sum_{P \in B_{P_i}} \frac{r_0(P)}{|P|} \]
PageRank

The Definition

\[ r(P) = \sum_{P \in B_P} \frac{r(P)}{|P|} \]

\( B_P = \{ \text{all pages pointing to } P \} \)

\( |P| = \text{number of out links from } P \)

Successive Refinement

Start with \( r_0(P_i) = \frac{1}{n} \) for all pages \( P_1, P_2, \ldots, P_n \)

Iteratively refine rankings for each page

\[ r_1(P_i) = \sum_{P \in B_{P_i}} \frac{r_0(P)}{|P|} \]

\[ r_2(P_i) = \sum_{P \in B_{P_i}} \frac{r_1(P)}{|P|} \]
PageRank

The Definition

\[ r(P) = \sum_{P \in B_P} \frac{r(P)}{|P|} \]

\( B_P = \{ \text{all pages pointing to} \ P \} \)

\( |P| = \text{number of out links from} \ P \)

Successive Refinement

Start with \( r_0(P_i) = 1/n \) for all pages \( P_1, P_2, \ldots, P_n \)

Iteratively refine rankings for each page

\[ r_1(P_i) = \sum_{P \in B_{P_i}} \frac{r_0(P)}{|P|} \]

\[ r_2(P_i) = \sum_{P \in B_{P_i}} \frac{r_1(P)}{|P|} \]

\[ \vdots \]

\[ r_{j+1}(P_i) = \sum_{P \in B_{P_i}} \frac{r_j(P)}{|P|} \]
In Matrix Notation

After Step $k$

$$\pi^T_k = [r_k(P_1), r_k(P_2), \ldots, r_k(P_n)]$$
In Matrix Notation

After Step $k$

$$\pi^T_k = [r_k(P_1), r_k(P_2), \cdots, r_k(P_n)]$$

$$\pi^T_{k+1} = \pi^T_k H$$

where

$$h_{ij} = \begin{cases} 
1/|P_i| & \text{if } i \rightarrow j \\
0 & \text{otherwise}
\end{cases}$$
In Matrix Notation

After Step $k$

$\pi^T_k = [r_k(P_1), r_k(P_2), \cdots, r_k(P_n)]$

$\pi^T_{k+1} = \pi^T_k H$ where $h_{ij} = \begin{cases} \frac{1}{|P_i|} & \text{if } i \rightarrow j \\ 0 & \text{otherwise} \end{cases}$

PageRank vector $= \pi^T = \lim_{k \to \infty} \pi^T_k$ is an eigenvector for $H$

Provided that the limit exists
Tiny Web

\[ H = \begin{pmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix} \]
Tiny Web

\[
H = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
\end{pmatrix}
\]
$$\begin{align*}
H &= \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{align*}$$
Tiny Web

\[ H = \begin{pmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1/3 & 1/3 & 0 & 0 & 0 & 1/3 \\ P_4 & & & & & & \\ P_5 & & & & & & \\ P_6 & & & & & & \end{pmatrix} \]
$$H = \begin{pmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ P_5 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_6 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$
$$H = \begin{pmatrix}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
\end{pmatrix}$$
$\mathbf{H} = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
\begin{array}{cccccc}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\end{pmatrix}$
Tiny Web

A random walk on the Web Graph

\[
H = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\
P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\
P_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\
P_6 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\]
Tiny Web

\[
\begin{bmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\
P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\
P_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\
P_6 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

A random walk on the Web Graph

PageRank = \( \pi_i = \) amount of time spent at \( P_i \)
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and quality of votes cast for that page.
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and quality of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and \textit{quality} of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and \textit{quality} of votes cast for that page.

Hyperlink as vote
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and quality of votes cast for that page.

Hyperlink as vote

page 2 is a dangling node
A random walk on the Web Graph

PageRank = $\pi_i = \text{amount of time spent at } P_i$

Dead end page (nothing to click on) — a “dangling node”
A random walk on the Web Graph

PageRank = $\pi_i = \text{amount of time spent at } P_i$

Dead end page (nothing to click on) — a “dangling node”

$\pi^T = (0, 1, 0, 0, 0, 0) = \text{e-vector} \implies \text{Page } P_2 \text{ is a “rank sink”}$
The Fix

Allow Web Surfers To Make Random Jumps
Ranking with a Random Surfer

- Rank each page corresponding to a search term by number and *quality* of votes cast for that page.

Hyperlink as vote

surfer “teleports”
The Fix

Allow Web Surfers To Make Random Jumps

- Replace zero rows with \( \frac{e^T}{n} = \left( \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right) \)

\[
S = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\
P_2 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\
P_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\
P_6 & 0 & 0 & 0 & 1 & 0 & 0 
\end{pmatrix}
\]
The Fix

Allow Web Surfers To Make Random Jumps

— Replace zero rows with \( \frac{e^T}{n} = \left( \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right) \)

\[
S = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
\]

— \( S = H + \frac{a e^T}{6} \) is now row stochastic \( \rho(S) = 1 \)
The Fix

Allow Web Surfers To Make Random Jumps

— Replace zero rows with \( \mathbf{e}^T = \left( \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right) \)

\[
\mathbf{s} = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{pmatrix}
\]

— \( \mathbf{s} = \mathbf{H} + \frac{\mathbf{a} \mathbf{e}^T}{6} \) is now row stochastic \( \implies \rho(\mathbf{s}) = 1 \)

— Perron says \( \exists \pi^T \succeq 0 \text{ s.t. } \pi^T = \pi^T \mathbf{s} \text{ with } \sum_i \pi_i = 1 \)
Nasty Problem

The Web Is Not Strongly Connected
Nasty Problem

The Web Is Not Strongly Connected

- S is reducible

\[
S = \begin{pmatrix}
P_1 & P_2 & P_3 & & & & \\
P_1 & 0 & 1/2 & 1/2 & & & 0 & 0 & 0 \\
P_2 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 \\
P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\
P_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 1/2 \\
P_6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}
\]
Nasty Problem

The Web Is Not Strongly Connected

- S is reducible

\[
S = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\
\frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
\]

- Reducible \implies \text{PageRank vector is not well defined}

- Frobenius says that \( S \) needs to be \textit{irreducible} to ensure a unique \( \pi^T > 0 \) s.t. \( \pi^T = \pi^T S \) with \( \sum_i \pi_i = 1 \)
Irreducibility Is Not Enough

Could Get Trapped Into A Cycle  \((P_i \rightarrow P_j \rightarrow P_i)\)
Irreducibility Is Not Enough

Could Get Trapped Into A Cycle \((P_i \rightarrow P_j \rightarrow P_i)\)

- The powers \(S^k\) fail to converge
Irreducibility Is Not Enough

Could Get Trapped Into A Cycle  \((P_i \rightarrow P_j \rightarrow P_i)\)

— The powers \(S^k\) fail to converge

— \(\pi_{k+1}^T = \pi_k^T S\) fails to convergence
Irreducibility Is Not Enough

Could Get Trapped Into A Cycle \((P_i \rightarrow P_j \rightarrow P_i)\)

- The powers \(S^k\) fail to converge
- \(\pi^{T}_{k+1} = \pi^T_k S\) fails to convergence

Convergence Requirement

- Perron–Frobenius requires \(S\) to be primitive
Irreducibility Is Not Enough

Could Get Trapped Into A Cycle \((P_i \rightarrow P_j \rightarrow P_i)\)

— The powers \(S^k\) fail to converge

— \(\pi_{k+1}^T = \pi_k^T S\) fails to converge

Convergence Requirement

— Perron–Frobenius requires \(S\) to be primitive

— No eigenvalues other than \(\lambda = 1\) on unit circle
Irreducibility Is Not Enough

Could Get Trapped Into A Cycle  \((P_i \rightarrow P_j \rightarrow P_i)\)

- The powers \(S^k\) fail to converge
- \(\pi_{k+1}^T = \pi_k^T S\) fails to convergence

Convergence Requirement

- Perron–Frobenius requires \(S\) to be primitive
- No eigenvalues other than \(\lambda = 1\) on unit circle
- Frobenius proved \(S\) is primitive  \(\iff S^k > 0\) for some \(k\)
The Google Fix

Allow A Random Jump From Any Page

\[ G = \alpha S + (1 - \alpha) E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \]
The Google Fix

Allow A Random Jump From Any Page

\[ G = \alpha S + (1 - \alpha)E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \]

\[ G = \alpha H + uv^T > 0 \quad u = \alpha a + (1 - \alpha)e, \quad v^T = e^T/n \]
The Google Fix

Allow A Random Jump From Any Page

\[ G = \alpha S + (1 - \alpha)E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \]

\[ G = \alpha H + uv^T > 0 \quad \quad \quad \quad \quad u = \alpha a + (1 - \alpha)e, \quad v^T = e^T/n \]

PageRank vector \[ \pi^T = \text{left-hand Perron vector of } G \]
Ranking with a Random Surfer

- If a page is “important,” it gets lots of votes from other important pages, which means the random surfer visits it often.

- Simply count the number of times, or proportion of time, the surfer spends on each page to create ranking of webpages.
Ranking with a Random Surfer

- If a page is “important,” it gets lots of votes from other important pages, which means the random surfer visits it often.

- Simply count the number of times, or proportion of time, the surfer spends on each page to create ranking of webpages.

Proportion of Time

Page 1 = .04
Page 2 = .05
Page 3 = .04
Page 4 = .38
Page 5 = .20
Page 6 = .29

Ranked List of Pages

Page 4
Page 6
Page 5
Page 2
Page 1
Page 3
The Google Fix

Allow A Random Jump From Any Page

\[ G = \alpha S + (1 - \alpha)E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \]

\[ G = \alpha H + uv^T > 0 \quad u = \alpha a + (1 - \alpha)e, \quad v^T = e^T/n \]

PageRank vector \( \pi^T = \text{left-hand Perron vector of } G \)

Some Happy Accidents

\[ x^T G = \alpha x^T H + \beta v^T \quad \text{Sparse computations with the original link structure} \]
The Google Fix

Allow A Random Jump From Any Page

\[ G = \alpha S + (1 - \alpha)E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \]

\[ G = \alpha H + uv^T > 0 \]

\[ u = \alpha a + (1 - \alpha)e, \quad v^T = e^T/n \]

PageRank vector

\[ \pi^T = \text{left-hand Perron vector of } G \]

Some Happy Accidents

\[ x^T G = \alpha x^T H + \beta v^T \]

Sparse computations with the original link structure

\[ \lambda_2(G) = \alpha \]

Convergence rate controllable by Google engineers
The Google Fix

Allow A Random Jump From Any Page

\[ G = \alpha S + (1 - \alpha)E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \]

\[ G = \alpha H + uv^T > 0 \quad u = \alpha a + (1 - \alpha)e, \quad v^T = e^T/n \]

PageRank vector \[ \pi^T = \text{left-hand Perron vector of } G \]

Some Happy Accidents

\[ x^T G = \alpha x^T H + \beta v^T \quad \text{Sparse computations with the original link structure} \]

\[ \lambda_2(G) = \alpha \quad \text{Convergence rate controllable by Google engineers} \]

\[ v^T \text{ can be any positive probability vector in } G = \alpha H + uv^T \]
The Google Fix

Allow A Random Jump From Any Page

- \( G = \alpha S + (1 - \alpha)E > 0, \quad E = ee^T/n, \quad 0 < \alpha < 1 \)
- \( G = \alpha H + uv^T > 0 \)

Some Happy Accidents

- \( x^T G = \alpha x^T H + \beta v^T \) \quad \text{Sparse computations with the original link structure}
- \( \lambda_2(G) = \alpha \) \quad \text{Convergence rate controllable by Google engineers}
- \( v^T \) can be any positive probability vector in \( G = \alpha H + uv^T \)
- The choice of \( v^T \) allows for personalization
Google matrix $G > 0$

Perron-Frobenius guarantees

• existence of $\pi$

• uniqueness of $\pi$

• convergence of algorithm

• rate of convergence: $\lambda_2(G) = \alpha$
Ranking Sports Teams
PageRank applied to Sports
(joint work with Carl Meyer)

- webpages vote with hyperlinks
- losing teams vote with point differentials

- not as successful at ranking and predicting winners as mHITS
mHITS

each team $i$ gets both offensive rating $o_i$ and defensive rating $d_i$

- **mHITS Thesis:** A team is a good defensive team (i.e., deserves a high defensive rating $d_j$) when it holds its opponents (particularly strong offensive teams) to low scores. A team is a good offensive team (i.e., deserves a high offensive rating $o_i$) when it scores many points against its opponents (particularly opponents with high defensive ratings).

\[
\begin{array}{c|ccccc}
 & Duke & Miami & UNC & UVA & VT \\
\hline
Duke & 0 & 7 & 21 & 7 & 0 \\
Miami & 52 & 0 & 34 & 25 & 27 \\
UNC & 24 & 16 & 0 & 7 & 3 \\
UVA & 38 & 17 & 5 & 0 & 14 \\
VT & 45 & 7 & 30 & 52 & 0 \\
\end{array}
\]

**Graph**

**Point Matrix** $P \geq 0$
mHITS Equations

Summation Notation

\[ d_j = \sum_{i \in I_j} p_{ij} \frac{1}{o_i} \quad \text{and} \quad o_i = \sum_{j \in L_i} p_{ij} \frac{1}{d_j} \]

Matrix Notation: iterative procedure

\[ d^{(k)} = P^T \frac{1}{o^{(k)}} \quad \text{and} \quad o^{(k)} = P \frac{1}{d^{(k-1)}} \]

- related to the Sinkhorn-Knopp algorithm for matrix balancing
  (uses successive row and column scaling to transform \( P \geq 0 \) into doubly stochastic matrix \( S \))

- P. Knight uses Sinkhorn-Knopp algorithm to rank webpages
mHITS Results: tiny NCAA
(data from Luke Ingram)

<table>
<thead>
<tr>
<th>Team</th>
<th>Off. Rating o</th>
<th>Off. Rank</th>
<th>Def. Rating d</th>
<th>Def. Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duke</td>
<td>.4</td>
<td>5th</td>
<td>140</td>
<td>5th</td>
</tr>
<tr>
<td>Miami</td>
<td>1.8</td>
<td>1st</td>
<td>67</td>
<td>2nd</td>
</tr>
<tr>
<td>UNC</td>
<td>.6</td>
<td>4th</td>
<td>97</td>
<td>4th</td>
</tr>
<tr>
<td>UVA</td>
<td>1.0</td>
<td>3rd</td>
<td>80</td>
<td>3rd</td>
</tr>
<tr>
<td>VT</td>
<td>1.4</td>
<td>2nd</td>
<td>34</td>
<td>1st</td>
</tr>
</tbody>
</table>

Below is a network diagram illustrating the relationships between the teams, with edge weights indicating the magnitudes of the ratings. The right table provides the $r = o/d$ rating and rank for each team:

<table>
<thead>
<tr>
<th>Team</th>
<th>r = o/d Rating</th>
<th>r = o/d Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duke</td>
<td>0.003</td>
<td>5th</td>
</tr>
<tr>
<td>Miami</td>
<td>0.027</td>
<td>2nd</td>
</tr>
<tr>
<td>UNC</td>
<td>0.006</td>
<td>4th</td>
</tr>
<tr>
<td>UVA</td>
<td>0.012</td>
<td>3rd</td>
</tr>
<tr>
<td>VT</td>
<td>0.041</td>
<td>1st</td>
</tr>
</tbody>
</table>
Weighted mHITS

- Weighted Point Matrix $\tilde{P} \geq 0$

  $$\tilde{p}_{ij} = w_{ij} \cdot p_{ij} \quad (w_{ij} = \text{weight of matchup between teams } i \text{ and } j)$$

- Possible weightings $w_{ij}$

---

Linear

Logarithmic

Exponential

Step Function
mHITS Results: full NCAA
(image from Neil Goodson and Colin Stephenson)
mHITS Results: full NCAA

- weightings can easily be applied to all ranking methods
  ⇒ interesting possibilities for weighted webpage ranking
mHITS Point Matrix \( P \geq 0 \)

Perron-Frobenius guarantees (for irreducible \( P \) with total support)

- existence of \( o \) and \( d \)
- uniqueness of \( o \) and \( d \)
- convergence of mHITS algorithm
- rate of convergence of mHITS algorithm

\[
\sigma_2^2(S), \quad \text{where} \quad S = D (1/o) \ P \ D (1/d)
\]
Recommendation Systems
Recommendation Systems

$A \geq 0$

$A = \begin{pmatrix}
1 & 5 & \ldots & 0 \\
0 & 0 & \ldots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 1 & \ldots & 2
\end{pmatrix}$

- Find similar users
- Find similar items
- Cluster users into classes

EX: use NMF so that $A_{m \times n} \approx W_{m \times k}H_{k \times n}$
**NMF Algorithm: Lee and Seung 2000**

**Mean Squared Error objective function**

\[
\min_{\mathbf{W}, \mathbf{H}} \| \mathbf{A} - \mathbf{WH} \|_F^2 \\
\text{s.t. } \mathbf{W}, \mathbf{H} \geq 0
\]

---

\[
\mathbf{W} = \text{abs}(\text{randn}(m,k)); \\
\mathbf{H} = \text{abs}(\text{randn}(k,n)); \\
\text{for } i = 1 : \text{maxiter} \\
\quad \mathbf{H} = \mathbf{H} \ast (\mathbf{W}^T \mathbf{A}) \div (\mathbf{W}^T \mathbf{WH} + 10^{-9}); \\
\quad \mathbf{W} = \mathbf{W} \ast (\mathbf{A} \mathbf{H}^T) \div (\mathbf{WH} \mathbf{H}^T + 10^{-9}); \\
\text{end}
\]

---

Many parameters affect performance (k, obj. function, sparsity constraints, algorithm, etc.).

— NMF is not unique!

(proof of convergence to fixed point based on E-M convergence proof)
Interpretation with NMF

- columns of $W$ are the underlying basis vectors, i.e., each of the $n$ columns of $A_{m \times n}$ can be built from $k$ columns of $W_{m \times k}$.

- columns of $H$ give the weights associated with each basis vector.

$$A_{*1} \approx WH_{*1} = \begin{bmatrix} \vdots & \vdots & \vdots \\ w_1 & h_{11} & + \vdots & w_2 & h_{21} & + \cdots & + & w_k & h_{k1} \end{bmatrix}$$

$A_{*2} = \begin{pmatrix} \text{item 1} & 5 \\ \text{item 2} & 0 \\ \text{item 3} & 0 \\ \text{item 4} & 11 \\ \text{item 5} & 1 \end{pmatrix} \approx WH_{*1} = \begin{bmatrix} 1.4 & 0 & 3.2 \\ 0 & 0 & 5.1 \\ 2.4 & .7 + 0 & .003 + \cdots + 0 & .2 \\ 0 & 4.4 \\ 0 & 2.8 \end{bmatrix}$

- $W, H \geq 0 \Rightarrow$ immediate interpretation (additive parts-based rep.)
Netflix

17770 movies, \approx 0.5 million users

\begin{align*}
&\text{user} \\
&\text{rating} \\
&\text{matrix} \\
A &= \begin{pmatrix}
\text{movie 1} & \text{User 1} & \text{User 2} & \ldots & \text{User n} \\
\text{movie 2} & 1 & 5 & \ldots & 0 \\
\vdots & 0 & 0 & \ddots & 1 \\
\text{movie m} & 0 & 1 & \ldots & 2
\end{pmatrix}
\end{align*}

- Find similar users
- Find similar movies
- Cluster users and movies into classes
- Rank users
- Rank movies
Clustering Netflix
(movie-movie matrix; 17770 x 17770)

,row [25];  Mystery Science Theater 3000: The Touch of Satan
.col [29];  29
.val [25][29] n/a

(vismatrix tool - David Gleich)
Netflix

17770 movies, ≈ .5 million users

\[ A = \begin{pmatrix}
\text{User 1} & \text{User 2} & \ldots & \text{User n} \\
\text{movie 1} & 1 & 5 & \ldots & 0 \\
\text{movie 2} & 0 & 0 & \ldots & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\text{movie m} & 0 & 1 & \ldots & 2 \\
\end{pmatrix} \]

- Find similar users
- Find similar movies
- Cluster users and movies into classes
- Rank users
- Rank movies
mHITS on Netflix

each movie $i$ gets a rating $m_i$ and each user gets a rating $u_j$

- mHITS Thesis: A movie is a good (i.e., deserves a high rating $m_i$) if it gets high ratings from good (i.e., discriminating) users. A user is good (i.e., deserves a high rating $u_j$) when his or her ratings match the true rating of a movie.

mHITS Netflix Algorithm

$$u = e;$$

for $i = 1 : \text{maxiter}$

$$m = A u;$$

$$m = \frac{5(m - \text{min}(m))}{\text{max}(m) - \text{min}(m)};$$

$$u = \frac{1}{((R - (R > 0).*(e*m^T)).^2)e};$$

end
Netflix mHITS results

subset: 1500 “super users”  (rate ≥ 1000 movies)

1st  Raiders of the Lost Ark
2nd  Silence of the Lambs
3rd  The Sixth Sense
4th  Shawshank Redemption
5th  LOR: Fellowship of the Ring
6th  The Matrix
7th  LOR: The Two Towers
8th  Pulp Fiction
9th  LOR: The Return of the King
10th  Forrest Gump
11th  The Usual Suspects
12th  American Beauty
13th  Pirates of the Carribbean: Black Pearl
14th  The Godfather
15th  Braveheart
The Enron Email Dataset

(data from Mike Berry)

- PRIVATE email collection of 150 Enron employees during 2001
- 92,000 terms and 65,000 messages
- Term-by-Message Matrix

\[
\begin{pmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{subpoena} & 2 & 0 & 1 & \ldots \\
\text{dynegy} & 0 & 3 & 0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
\]
Text Mining Applications

- Data compression
- Find similar terms
- Find similar documents
- Cluster documents
- Cluster terms
- Topic detection and tracking
Unclustered Enron A
NMF-clustered Enron A
## Clustering the Enron Email Dataset
(image from Mike Berry)

<table>
<thead>
<tr>
<th>Feature Index (k)</th>
<th>Cluster Size</th>
<th>Topic Description</th>
<th>Dominant Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>497</td>
<td>California</td>
<td>ca, epuc, gov, socaigas, sempra, org, see, gmssr, aelaw, ci</td>
</tr>
<tr>
<td>23</td>
<td>43</td>
<td>Louise Kitchen named top woman by Fortune</td>
<td>cvp, fortune, britain, woman, ceo, avon, fiorinal, cfo, hewlett, packard</td>
</tr>
<tr>
<td>26</td>
<td>231</td>
<td>Fantasy football</td>
<td>game, wr, qb, play, rb, season, injury, updated, fantasy, image</td>
</tr>
<tr>
<td>33</td>
<td>233</td>
<td>Texas longhorn football newsletter</td>
<td>UT, orange, longhorn[s], texas, true, truorange, recruiting, oklahoma defensive</td>
</tr>
<tr>
<td>34</td>
<td>65</td>
<td>Enron collapse</td>
<td>partnership[s], fastow, shares, sec, stock, shareholder, investors, equity, lay</td>
</tr>
<tr>
<td>39</td>
<td>235</td>
<td>Emails about India</td>
<td>dahhol, dpc, india, msea, maharashtra, indian, lenders, delhi, foreign, minister</td>
</tr>
<tr>
<td>46</td>
<td>127</td>
<td>Enron collapse</td>
<td>dow, debt, reserved, wall, copyright jones, cents, analysts, reuters, spokesman</td>
</tr>
</tbody>
</table>
Tracking Enron clusters over time
(image from Mike Berry)
Meta-Algorithms
Rank Aggregation
Rank Aggregation

- average rank
- Borda count
- simulated data
- graph theory
### Average Rank

<table>
<thead>
<tr>
<th></th>
<th>mHITS ($r = o/d$)</th>
<th>Massey</th>
<th>Colley</th>
<th>Average Rating</th>
<th>Average Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duke</td>
<td>5th</td>
<td>5th</td>
<td>5th</td>
<td>5</td>
<td>5th</td>
</tr>
<tr>
<td>Miami</td>
<td>2nd</td>
<td>1st</td>
<td>1st</td>
<td>1.3</td>
<td>1st</td>
</tr>
<tr>
<td>UNC</td>
<td>4th</td>
<td>4th</td>
<td>3rd</td>
<td>3.6</td>
<td>4th</td>
</tr>
<tr>
<td>UVA</td>
<td>3rd</td>
<td>3rd</td>
<td>4th</td>
<td>3.3</td>
<td>3rd</td>
</tr>
<tr>
<td>VT</td>
<td>1st</td>
<td>2nd</td>
<td>2nd</td>
<td>1.6</td>
<td>2nd</td>
</tr>
</tbody>
</table>
Borda Count

- for each ranked list, each item receives a score equal to the number of items it outranks.

<table>
<thead>
<tr>
<th></th>
<th>mHITS (r = o/d)</th>
<th>Massey</th>
<th>Colley</th>
<th>Borda Count</th>
<th>Borda Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duke</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5th</td>
</tr>
<tr>
<td>Miami</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>11</td>
<td>1st</td>
</tr>
<tr>
<td>UNC</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4th</td>
</tr>
<tr>
<td>UVA</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3rd</td>
</tr>
<tr>
<td>VT</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>2nd</td>
</tr>
</tbody>
</table>
Simulated Data

3 ranked lists

<table>
<thead>
<tr>
<th>mHITS</th>
<th>Massey</th>
<th>Colley</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st VT</td>
<td>1st Miami</td>
<td>1st Miami</td>
</tr>
<tr>
<td>2nd Miami</td>
<td>2nd VT</td>
<td></td>
</tr>
<tr>
<td>3rd UVA</td>
<td>3rd UVA</td>
<td></td>
</tr>
<tr>
<td>4th UNC</td>
<td>4th UNC</td>
<td></td>
</tr>
<tr>
<td>5th Duke</td>
<td>5th Duke</td>
<td></td>
</tr>
</tbody>
</table>

mHITS

- VT beats Miami by 1 point, UVA by 2 points, . . .
- Miami beats UVA by 1 point, UNC by 2 points, . . .
- UVA beats UNC by 1 point, Duke by 2 points
- UNC beats Duke by 1 point

repeat for each ranked list ⇒ generates game scores for teams

Simulated Game Data
Simulated Data

Ranked Lists from Various Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Team 1</th>
<th>Team 2</th>
<th>...</th>
<th>Team n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Markov</td>
<td>7-4</td>
<td>17-8</td>
<td></td>
<td>6-21</td>
</tr>
<tr>
<td></td>
<td>1-5</td>
<td>4-2</td>
<td></td>
<td>12-10</td>
</tr>
<tr>
<td>Colley</td>
<td>3-11</td>
<td>6-8</td>
<td>13-19</td>
<td>9-12</td>
</tr>
<tr>
<td>mH/TS</td>
<td>1-3</td>
<td>15-21</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulated Game Data

that becomes input to

Combiner Method

that creates

Aggregated List
Graph Theory

more voting

- ranked lists are used to form weighted graph
- possible weights
  * \( w_{ij} = \# \) of ranked lists having \( i \) below \( j \)
  * \( w_{ij} = \) sum of rank differences of lists having \( i \) below \( j \)

- run algorithm (e.g., Markov, PageRank, HITS) to determine most important nodes
Aggregation

Rating Aggregation
Rating Aggregation

rating vectors

- form rating differential matrix $R$ for each rating vector
Rating Aggregation

describing differential matrices \( R \geq 0 \)

- differing scales \( \rightarrow \) NORMALIZE

\[
R_{\text{Massey}} = \begin{pmatrix}
Duke & Miami & UNC & UVA & VT \\
Duke & 0 & 0 & 0 & 0 & 0 \\
Miami & 0 & 26.2 & 21.6 & 0.2 & 0 \\
UNC & 0 & 0 & 0 & 0 & 0 \\
UVA & 0 & 4.6 & 0 & 0 & 0 \\
VT & 0 & 26 & 21.4 & 0 & 0
\end{pmatrix}
\]

\[
R_{\text{Colley}} = \begin{pmatrix}
Duke & Miami & UNC & UVA & VT \\
Duke & 0 & 0 & 0 & 0 & 0 \\
Miami & 0 & 0 & 0 & 0 & 0 \\
UNC & 0 & 0 & 0 & 0 & 0 \\
UVA & 0 & 0 & 0 & 0 & 0 \\
VT & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
R_{mHITS} = \begin{pmatrix}
Duke & Miami & UNC & UVA & VT \\
Duke & 0 & 0 & 0 & 0 & 0 \\
Miami & 0 & 0 & 0 & 0 & 0 \\
UNC & 0 & 0 & 0 & 0 & 0 \\
UVA & 0 & 0 & 0 & 0 & 0 \\
VT & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
Rating Aggregation

rating differential matrices
Rating Aggregation

rating differential matrices

- combine into one matrix → AVERAGE
Rating Aggregation

rating differential matrices

- combine into one matrix → AVERAGE
Rating Aggregation

average rating differential matrix $R_{average} \geq 0$

- run ranking method
  - Markov method on $R_{average}^T$
  - row sums of $R_{average}$ / col sums of $R_{average}$
  - Perron vector of $R_{average}$
Rating Aggregation

average rating differential matrix $\mathbf{R}_{\text{average}} \geq 0$

- run ranking method
  - Markov method on $\mathbf{R}_{\text{average}}^T$
  - row sums of $\mathbf{R}_{\text{average}}$ / col sums of $\mathbf{R}_{\text{average}}$
  - Perron vector of $\mathbf{R}_{\text{average}}$

<table>
<thead>
<tr>
<th>Team</th>
<th>Method 1 $r = o/d$</th>
<th>Method 2 Markov $r$</th>
<th>Method 3 Perron $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duke</td>
<td>0</td>
<td>.020 $5^{th}$</td>
<td>.27 $5^{th}$</td>
</tr>
<tr>
<td>Miami</td>
<td>16.4 $2^{nd}$</td>
<td>.465 $2^{nd}$</td>
<td>.58 $2^{nd}$</td>
</tr>
<tr>
<td>UNC</td>
<td>.4 $3^{rd}$</td>
<td>.025 $3^{rd}$</td>
<td>.34 $3^{rd}$</td>
</tr>
<tr>
<td>UVA</td>
<td>3 $4^{th}$</td>
<td>.024 $4^{th}$</td>
<td>.33 $4^{th}$</td>
</tr>
<tr>
<td>VT</td>
<td>26.0 $1^{st}$</td>
<td>.466 $1^{st}$</td>
<td>.61 $1^{st}$</td>
</tr>
</tbody>
</table>
Cluster Aggregation

many clustering algorithms = many clustering results

⇒ Can we combine many results to make one super result?

Cluster Aggregation Algorithm

1. Create aggregation matrix $F \geq 0$

   $f_{ij} = \# \text{ of methods having items } i \text{ and } j$
   \hspace{1cm} in the same cluster

2. Run favorite clustering method on $F$
**Cluster Aggregation Example**

<table>
<thead>
<tr>
<th>Method 1</th>
<th>Method 2</th>
<th>Method 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>cluster assignment</td>
<td>item</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

**Cluster Aggregated Results**

- **Fiedler using just one eigenvector**
  - 1 4 5
  - 2 3 6

- **Fiedler using two eigenvectors**
  - 1 4
  - 3 6
  - 5
  - 2

**Cluster Aggregated Graph**

1 2 3
4 5 6

1 1 1
3 3 3
Conclusions

- Applied problems often have nonnegative data.
- This nonnegativity is exploitable structure.
  ⇒ proofs for convergence and rates of convergence
  ⇒ proofs for existence and uniqueness
- Nonnegativity is easily interpretable. Often it pays to maintain nonnegativity throughout the modeling process.
  ⇒ nonnegative matrix factorizations
  ⇒ nonnegative rating vectors