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
- Indexing Module
- Query Module
- Ranking Module
- User
- Queries
- Results
- Indexes
- 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 .com :: 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 \( Y \)
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 $Y$
PageRank

The Definition

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

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

\[ |P| = \text{number of out links from } P \]
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 \( P_1, P_2, \ldots, P_n \)
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

\[ r(P) = \sum_{P \in \mathcal{B}_P} \frac{r(P)}{|P|} \]

\( \mathcal{B}_P = \{ \text{all pages pointing to } P \} \)

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

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

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

\[ r_2(P_i) = \sum_{P \in \mathcal{B}_{P_i}} \frac{r_1(P)}{|P|} \]
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 $k$

$$\pi_k^T = [r_k(P_1), r_k(P_2), \cdots, r_k(P_n)]$$
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 $h_{ij} = \begin{cases} 1/|P_i| & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$
In Matrix Notation

After Step $k$

$- \quad \pi_k^T = [r_k(P_1), r_k(P_2), \cdots, r_k(P_n)]$

$- \quad \pi_{k+1}^T = \pi_k^T H$ \quad where \quad $h_{ij} = \begin{cases} 1/|P_i| & \text{if } i \rightarrow j \\ 0 & \text{otherwise} \end{cases}$

$- \quad \text{PageRank vector} = \pi^T = \lim_{k \to \infty} \pi_k^T = \text{eigenvector for } H$

Provided that the limit exists
Tiny Web

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

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

\[ \begin{pmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ P_1 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1/3 & 1/3 & 0 & 0 & 0 & 1/3 \\ P_4 & 0 & 0 & 0 & 0 & 1/3 & 0 \end{pmatrix} \]
$$H = \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} \frac{1}{2}
\end{pmatrix}$$
\[
\begin{align*}
H &= \begin{pmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
\end{pmatrix}
\end{align*}
\]

Tiny Web
Tiny Web

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

A random walk on the Web Graph
A random walk on the Web Graph

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

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 *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.
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

page 2 is a dangling node
\[ H = \begin{pmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ P_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ P_6 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \]

\[ \text{Tiny Web} \]

\[ \text{A random walk on the Web Graph} \]

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

\[ \text{Dead end page (nothing to click on) — a “dangling node”} \]
A random walk on the Web Graph

PageRank = \( \pi_i \) = amount of time spent at \( P_i \)

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

\( \pi^T = (0, 1, 0, 0, 0, 0) = \) e-vector \( \implies \) Page \( P_2 \) 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 $\frac{e^T}{n} = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$

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

Allow Web Surfers To Make Random Jumps

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

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

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

Allow Web Surfers To Make Random Jumps

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

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

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

— Perron says \( \exists \pi^T \geq 0 \) s.t. \( \pi^T = \pi^T S \) 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

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

The Web Is Not Strongly Connected

- S is reducible

\[
S = \begin{pmatrix}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
1/6 & 1/6 & 1/6 & 1/6 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 0 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
\]

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

- Frobenius says 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

- \(\pi^T_{k+1} = \pi^T_k S\) 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
- \(\pi_{k+1}^T = \pi_k^T S\) 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 convergence

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 \((P_i \to P_j \to P_i)\)

- 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

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

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

Allow A Random Jump From Any Page

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

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

- PageRank vector \( \pi^T = \) 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.

<table>
<thead>
<tr>
<th>Proportion of Time</th>
<th>Ranked List of Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 1 = .04</td>
<td>Page 4</td>
</tr>
<tr>
<td>Page 2 = .05</td>
<td>Page 6</td>
</tr>
<tr>
<td>Page 3 = .04</td>
<td>Page 5</td>
</tr>
<tr>
<td>Page 4 = .38</td>
<td>Page 2</td>
</tr>
<tr>
<td>Page 5 = .20</td>
<td>Page 1</td>
</tr>
<tr>
<td>Page 6 = .29</td>
<td>Page 3</td>
</tr>
</tbody>
</table>
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

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

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

PageRank vector \( \pi^T = \) 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

- \( v^T \) can be any positive probability vector in \( G = \alpha H + uv^T \)
**The Google Fix**

**Allow A Random Jump From Any Page**

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

- $G = \alpha H + uv^T > 0$

- 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

- $v^T$ can be any positive probability vector in $G = \alpha H + uv^T$

- The choice of $v^T$ allows for personalization
PageRank Issues

- Computing PageRank: simulation, eigensystem, linear system; accuracy
- power law distribution: sensitivity, spamming
- link strategies
- overuse