The PageRank Algorithm
Why PageRank?

• Suppose you have a directed graph
  - Websites linking to one another
  - Recommendation votes on eBay, or AirBnB, NetFlix, etc.
  - Scientists referring to each other’s works
  - Neighborhoods in cities connected by movement of pedestrians
  - Recommendation for leadership of communities
• How do you associate a good “popularity” or “rank” value to each node in the graph?
• This is what the PageRank Algorithm is about.
PageRank

- The PageRank Algorithm as invented by Larry Page in 1998 when he was a graduate student at Stanford
- He started a research project called “BackRub”
- Sergey Brin joined the project pretty much right away
- They went on to write the paper on the right.
- Goal was to “bring order into the Web”

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department, Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.
Inventor

- Larry Page patented the procedure
  - US Patent 6,285,999
  - Filed Jan 9, 1998
  - Granted Sep 4, 2001
- Owner is Stanford University
- Probably one of the most lucrative patents of all times
Directed Graphs

• A directed graph is a set $V$ of vertices and a set $E$ of edges, $E \subseteq V \times V$.
  - $(u,v) \in E$ connects vertices $u, v \in V$.
  - $u$ is the starting point and $v$ the endpoint of the edge.

• A directed graph on a set $V$ is also called a relation on $V$. 
Example of a Directed Graph

Vertices:
Websites

Relationship:
Directed edge between website A and website B if there is a link from website A to website B
Degrees (again)

- The *in-degree* \( \text{deg}^-(v) \) of a node \( v \) is the number of edges ending in the node; the *out-degree* \( \text{deg}^+(v) \) is the number of edges starting at the node.
- Formally:
  - \( \text{deg}^+(u) = |\{(u,v) \in E\}| \)
  - \( \text{deg}^-(u) = |\{(v,u) \in E\}| \)
Adjacency Matrix

- $G = (V,E)$ directed graph, $V=\{v_1,\ldots,v_n\}$. An adjacency matrix for $G$ is an $n \times n$-matrix $A=(a_{ij})$ such that
  - $a_{ij} = 1$ if $(v_i,v_j) \in E$, and $a_{ij} = 0$ otherwise.
- Note that the adjacency matrix depends on the ordering of the elements of $V$ (hence is not unique).

Sum of entries in row $i$ is the out-degree of node $v_i$

Matrix is not symmetric in general

Sum of entries in column $i$ is the in-degree of node $v_i$
Back to PageRank: Example
First Idea

Use the in-degree as a measure of popularity

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>E</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B wins the popularity contest
Really that Good?

- No.
- Can be very easily rigged.
Can we do Better?

- But if B is popular, and B is pointing to C, then C should also be popular.
- But then D should also be popular, since C is popular and thinks that D is popular as well.
A Different Way: Continuous Voting

- Distribute a fixed number of votes to every player at the start

- In every round, each player takes its votes, and gives them in an equal fashion to all the other players it is voting for
  - So, for example, if it is pointing to two other players, then half of its votes go to one, the other half to the other

- Run this for as long as it takes

- Hopefully, after a few rounds the number of votes of every player stays almost the same

- That number can be a measure of popularity
Example

A \quad E/3
B \quad A/2 + C/2 + D + E/3
C \quad B + E/3
D \quad C/2
E \quad A/2

<table>
<thead>
<tr>
<th></th>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
<th>Round 4</th>
<th>Round 5</th>
<th>Round 6</th>
<th>Round 7</th>
<th>Round 8</th>
<th>Round 9</th>
<th>Round 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.200</td>
<td>0.067</td>
<td>0.033</td>
<td>0.011</td>
<td>0.006</td>
<td>0.002</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>B</td>
<td>0.200</td>
<td>0.467</td>
<td>0.300</td>
<td>0.411</td>
<td>0.417</td>
<td>0.369</td>
<td>0.419</td>
<td>0.395</td>
<td>0.395</td>
<td>0.407</td>
</tr>
<tr>
<td>C</td>
<td>0.200</td>
<td>0.267</td>
<td>0.500</td>
<td>0.311</td>
<td>0.417</td>
<td>0.419</td>
<td>0.369</td>
<td>0.420</td>
<td>0.395</td>
<td>0.395</td>
</tr>
<tr>
<td>D</td>
<td>0.200</td>
<td>0.100</td>
<td>0.133</td>
<td>0.250</td>
<td>0.156</td>
<td>0.208</td>
<td>0.209</td>
<td>0.185</td>
<td>0.210</td>
<td>0.197</td>
</tr>
<tr>
<td>E</td>
<td>0.200</td>
<td>0.100</td>
<td>0.033</td>
<td>0.017</td>
<td>0.006</td>
<td>0.003</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Adjacency Matrix Form

\[
\begin{array}{cccccc}
A & B & C & D & E & \text{Term} \\
0 & 0 & 0 & 0 & 0 & A/2+C/2+D+E/3 \\
1/2 & 0 & 1/2 & 1 & 1/3 & B+E/3 \\
0 & 1 & 0 & 0 & 1/3 & C/2 \\
0 & 0 & 1/2 & 0 & 0 & A/2 \\
1/2 & 0 & 0 & 0 & 0 & A/3
\end{array}
\]

\[
\left[
\begin{array}{cccccc}
A & B & C & D & E \\
0 & 0 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 1 & 1/3 \\
0 & 1 & 0 & 0 & 1/3 \\
0 & 0 & 1/2 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 0
\end{array}
\right] \times
\left[
\begin{array}{c}
\text{A} \\
\text{B} \\
\text{C} \\
\text{D} \\
\text{E}
\end{array}
\right] =
\left[
\begin{array}{c}
\text{E}/3 \\
\text{A}/2+C/2+D+E/3 \\
\text{B}+\text{E}/3 \\
\text{C}/2 \\
\text{A}/2
\end{array}
\right]
\]
Recursion

\[ v_0 = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix} \quad v_{k+1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1/3 \\ 1/2 & 0 & 1/2 & 1 & 1/3 \\ 0 & 1 & 0 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot v_k =: A \cdot v_k \]

Does this recursion converge to a fixed point?

\[ v_k = A^k \cdot v_0 \]
**Diagonalization**

\[
A = \begin{pmatrix}
0 & 0 & 0 & 0 & 1/3 \\
1/2 & 0 & 1/2 & 1 & 1/3 \\
0 & 1 & 0 & 0 & 1/3 \\
0 & 0 & 1/2 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 0 \\
\end{pmatrix} = T \cdot \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & -1/2 + i/2 & 0 & 0 & 0 \\
0 & 0 & 1/2 - i/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \cdot T^{-1}
\]

Absolute value of these eigenvalues is < 1.

\[
A^k \rightarrow T \cdot \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \cdot T^{-1}
\]

**Recursion converges!**
How to Find the Solution

Fixed point \( w \) satisfies \( w = A \cdot w \)

So, \( w \) is an eigenvector with eigenvalue \( 1 = \)

Vector unique subject to sum of entries = 1

<table>
<thead>
<tr>
<th></th>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
<th>Round 4</th>
<th>Round 5</th>
<th>Round 6</th>
<th>Round 7</th>
<th>Round 8</th>
<th>Round 9</th>
<th>Round 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.200</td>
<td>0.067</td>
<td>0.033</td>
<td>0.011</td>
<td>0.006</td>
<td>0.002</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>B</td>
<td>0.200</td>
<td>0.467</td>
<td>0.300</td>
<td>0.411</td>
<td>0.417</td>
<td>0.369</td>
<td>0.419</td>
<td>0.395</td>
<td>0.395</td>
<td>0.407</td>
</tr>
<tr>
<td>C</td>
<td>0.200</td>
<td>0.267</td>
<td>0.500</td>
<td>0.311</td>
<td>0.417</td>
<td>0.419</td>
<td>0.369</td>
<td>0.420</td>
<td>0.395</td>
<td>0.395</td>
</tr>
<tr>
<td>D</td>
<td>0.200</td>
<td>0.100</td>
<td>0.133</td>
<td>0.250</td>
<td>0.156</td>
<td>0.208</td>
<td>0.209</td>
<td>0.185</td>
<td>0.210</td>
<td>0.197</td>
</tr>
<tr>
<td>E</td>
<td>0.200</td>
<td>0.100</td>
<td>0.033</td>
<td>0.017</td>
<td>0.006</td>
<td>0.003</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Rigging

- Same eigenvector for eigenvalue 1
- Rigging would not work
Cooperative Rigging

Graph showing connections between nodes A, B, C, D, and E with associated probabilities.

Matrix:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bar chart showing distribution: 40% for B and C, 20% for D.
Cooperative Rigging

![Cooperative Rigging Diagram]

The diagram illustrates the cooperative rigging relationships among different entities (A, B, C, D, E). The percentages and values are as follows:

- A: 18.18%
- B: 45.46%
- C: 9.1%
- D: 9.1%
- E: 9.1%

The table shows the transition probabilities:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>0</td>
<td>1/5</td>
<td>1</td>
<td>1/3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The graph shows the alteration (red) and before (blue) percentages for A, B, C, D, and E.
Perron-Frobenius Theorem

- Theorem about the eigenvectors and eigenvalues of “non-negative” matrices
  - First proved by Perron for “positive” matrices in 1907
    ➡ Matrices having strictly positive entries
  - Later generalized by Frobenius to non-negative matrices of a particular type in 1912
    ➡ Matrices having non-negative entries
    ➡ Such that the underlying directed graph is strongly connected
Definitions

• A matrix is called non-negative if all of its entries are $\geq 0$
• A matrix is called irreducible if for any of its entries $(i,j)$ there is a $k$ such that the $(i,j)$-entry of $A^k$ is positive.
  - This means that the underlying directed graph is strongly connected
  ➔ This means that for any two nodes in the graph there is a directed path connecting them
Perron-Frobenius Theorem (Abridged Version)

- A non-negative irreducible matrix
- Then $A$ has a positive (real) eigenvalue $\lambda_{\text{max}}$ and for all other eigenvalues $\lambda$ we have

$$|\lambda| \leq \lambda_{\text{max}}$$

Moreover, if the sum of the entries of the columns of $A$ is 1 for every column, then $\lambda_{\text{max}} = 1$

This last part is a corollary and not really a part of the theorem

- The theorem can be used to prove convergence of the iteration

Caveat: the matrices we obtain are not always irreducible
PageRank

- **Basic Idea: Taxation**
  - Imagine the votes being money transferred from one node to another
  - At every iteration, the amount of money at each node is taxed at the rate of $t < 1$.
  - The money raised this way is equally distributed among all the nodes in the graph for the next iteration.
PageRank

• What does it mean for websites?
  - For websites: if people start clicking on outgoing links, then at each stage they have a certain probability of getting bored and moving to another random webpage
    ➡ Typical tax rate is 15%

• What does it mean for payments or votes?
  - Through taxation, even unpopular members can have some chance of survival
    ➡ Tax rate should depend on the preferred outcome
PageRank

Tax rate: 15%

Distribute the tax equally among the nodes.
PageRank: Mathematical Formulation

\[ N = \text{total number of nodes} \]

\[ v_0 = \begin{pmatrix} 1/N \\ 1/N \\ \vdots \\ 1/N \end{pmatrix} \]

At the beginning:
All nodes receive equal votes

At the next iteration:
\[ v_{k+1} = (1 - t) A \cdot v_k + \begin{pmatrix} t/N \\ t/N \\ \vdots \\ t/N \end{pmatrix} \]

Tax rate: \( t \)

Distribute the tax equally among the nodes
Fixed Point $w$

$$w = (1 - t)Aw + tv_0$$

$w = t(I - (1 - t)A)^{-1} \cdot v_0$

Convergence guaranteed by Perron-Frobenius
Example

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

\[ (I - 0.85 \times A)^{-1} = \begin{pmatrix} 1.13690194220748 & -0. & -0. & -0. & 0.322122216958787 \\ 2.1940616455640 & 3.01488599962314 & 2.3704541720369 & 2.5626309967967 & 2.14748144639191 \\ 2.001807118208042 & 2.56265309967967 & 3.01488599962314 & 2.17825513472772 & 2.14748144639191 \\ 0.839078052384179 & 1.08912756736386 & 1.28132654983983 & 1.92575843225928 & 0.912679614716564 \\ 0.43183325438181 & 0. & 0. & 0. & 1.13690194220748 \end{pmatrix} \]

\[ W = \begin{pmatrix} 0.0438 & 0.3687 & 0.3572 & 0.1818 & 0.0486 \end{pmatrix} \]
Rigging is still possible
Rigging

- Cooperative rigging becomes exceedingly difficult (but not impossible) as the graph grows
  - Only a small part of the graph is modified
- but other countermeasures are needed
Implementation

• In reality, we don’t compute eigenvectors of matrices or their inverses
• Computation is done via “simulation” or “iteration”
• If the eigenvalues of the matrix are small, then iteration can converge quickly to desired solution