All web pages are not equally “important”

www.joe-schmoe.com vs. www.stanford.edu

There is large diversity in the web-graph node connectivity.

Let’s rank the pages by the link structure!
PageRank: The “Flow” Formulation
Idea: Links as votes

- Page is more important if it has more links
  - In-coming links? Out-going links?

Think of in-links as votes:

- www.stanford.edu has 23,400 in-links
- www.joe-schmoe.com has 1 in-link

Are all in-links are equal?

- Links from important pages count more
- Recursive question!

Each link’s vote is proportional to the importance of its source page.

If page $j$ with importance $r_j$ has $n$ out-links, each link gets $r_j/n$ votes.

Page $j$’s own importance is the sum of the votes on its in-links:

$$r_j = r_i/3 + r_k/4$$
A “vote” from an important page is worth more

A page is important if it is pointed to by other important pages

Define a “rank” $r_j$ for page $j$

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

di ... out-degree of node $i$

"Flow" equations:

$$r_y = \frac{r_y}{2} + \frac{r_a}{2}$$
$$r_a = \frac{r_y}{2} + r_m$$
$$r_m = \frac{r_a}{2}$$

The web in 1839
Solving the Flow Equations

- 3 equations, 3 unknowns, no constants
  - No unique solution
  - All solutions equivalent modulo the scale factor

- Additional constraint forces uniqueness:
  - \( r_y + r_a + r_m = 1 \)
  - Solution: \( r_y = \frac{2}{5}, \ r_a = \frac{2}{5}, \ r_m = \frac{1}{5} \)

- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs

- We need a new formulation!
PageRank: Matrix Formulation

- **Stochastic adjacency matrix** $M$
  - Let page $i$ has $d_i$ out-links
  - If $i \rightarrow j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
    - $M$ is a **column stochastic matrix**
      - Columns sum to 1
- **Rank vector** $r$: vector with an entry per page
  - $r_i$ is the importance score of page $i$
  - $\sum_i r_i = 1$
- The flow equations can be written
  $$r = M \cdot r$$
Example

- Remember the flow equation:
- Flow equation in the matrix form
  \[ M \cdot r = r \]
- Suppose page \( i \) links to 3 pages, including \( j \)

\[
\begin{array}{c}
M \\
\cdot \\
r \\
= \\
r
\end{array}
\]

Eigenvector Formulation

- The flow equations can be written
  \[ r = M \cdot r \]

- So the rank vector \( r \) is an eigenvector of the stochastic web matrix \( M \)
  - In fact, its first or principal eigenvector, with corresponding eigenvalue \( 1 \)
    - Largest eigenvalue of \( M \) is \( 1 \) since \( M \) is column stochastic (with non-negative entries)
      - We know \( r \) is unit length and each column of \( M \) sums to one, so \( Mr \leq 1 \)

- We can now efficiently solve for \( r! \)
  The method is called Power iteration

NOTE: \( x \) is an eigenvector with the corresponding eigenvalue \( \lambda \) if:
\[ Ax = \lambda x \]
**Example: Flow Equations & M**

\[ r = M \cdot r \]

\[
\begin{align*}
    r_y &= r_y / 2 + r_a / 2 \\
    r_a &= r_y / 2 + r_m \\
    r_m &= r_a / 2
\end{align*}
\]

\[
\begin{bmatrix}
    y \\
    a \\
    m
\end{bmatrix}
= 
\begin{bmatrix}
    \frac{1}{2} & \frac{1}{2} & 0 \\
    \frac{1}{2} & 0 & 1 \\
    0 & \frac{1}{2} & 0
\end{bmatrix}
\begin{bmatrix}
    y \\
    a \\
    m
\end{bmatrix}
\]
Given a web graph with \( n \) nodes, where the nodes are pages and edges are hyperlinks.

**Power iteration:** a simple iterative scheme

- Suppose there are \( N \) web pages.
- Initialize: \( r^{(0)} = [1/N, \ldots, 1/N]^T \)
- Iterate: \( r^{(t+1)} = M \cdot r^{(t)} \)
- Stop when \( |r^{(t+1)} - r^{(t)}|_1 < \varepsilon \)

\[ |x|_1 = \sum_{1 \leq i \leq N} |x_i| \] is the \( L_1 \) norm. Can use any other vector norm, e.g., Euclidean.
PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - **1:** $r'_j = \sum_{i\to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - Goto 1

- **Example:**

\[
\begin{pmatrix}
  r_y \\
r_a \\
r_m
\end{pmatrix} = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}
\]

Iteration 0, 1, 2, …
PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - $1: r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - $2: r = r'$
  - Goto 1

- **Example:**

\[
\begin{pmatrix}
    r_y \\
r_a \\
r_m
\end{pmatrix} =
\begin{pmatrix}
    1/3 & 1/3 & 5/12 & 9/24 & 6/15 \\
    1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\
    1/3 & 1/6 & 3/12 & 1/6 & 3/15
\end{pmatrix}
\]

Iteration 0, 1, 2, …

\[
\begin{array}{ccc}
    y & a & m \\
    \frac{1}{2} & \frac{1}{2} & 0 \\
    \frac{1}{2} & 0 & 1 \\
    0 & \frac{1}{2} & 0 \\
\end{array}
\]

\[
\begin{align*}
    r_y &= \frac{r_y}{2} + \frac{r_a}{2} \\
    r_a &= \frac{r_y}{2} + \frac{r_m}{2} \\
    r_m &= \frac{r_a}{2}
\end{align*}
\]
Why Power Iteration works? (1)

- **Power iteration:**
  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - $r^{(1)} = M \cdot r^{(0)}$
  - $r^{(2)} = M \cdot r^{(1)} = M(Mr^{(1)}) = M^2 \cdot r^{(0)}$
  - $r^{(3)} = M \cdot r^{(2)} = M(M^2r^{(0)}) = M^3 \cdot r^{(0)}$

- **Claim:**
  Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, ... M^k \cdot r^{(0)}, ...$ approaches the dominant eigenvector of $M$
Why Power Iteration works? (2)

- **Claim:** Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \ldots M^k \cdot r^{(0)}, \ldots$ approaches the dominant eigenvector of $M$

- **Proof:**
  - Assume $M$ has $n$ linearly independent eigenvectors, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_1 > \lambda_2 > \ldots > \lambda_n$
  - Vectors $x_1, x_2, \ldots, x_n$ form a basis and thus we can write: $r^{(0)} = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$
  - $M r^{(0)} = M(c_1 x_1 + c_2 x_2 + \cdots + c_n x_n)$
    
    $= c_1 (M x_1) + c_2 (M x_2) + \cdots + c_n (M x_n)$
    
    $= c_1 (\lambda_1 x_1) + c_2 (\lambda_2 x_2) + \cdots + c_n (\lambda_n x_n)$
  - Repeated multiplication on both sides produces $M^k r^{(0)} = c_1 (\lambda_1^k x_1) + c_2 (\lambda_2^k x_2) + \cdots + c_n (\lambda_n^k x_n)$
Why Power Iteration works? (3)

- **Claim:** Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \ldots M^k \cdot r^{(0)}, \ldots$ approaches the dominant eigenvector of $M$

- **Proof (continued):**
  - Repeated multiplication on both sides produces
    
    $$M^k r^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \cdots + c_n(\lambda_n^k x_n)$$

    $$= \lambda_1^k \left[ c_1 x_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \cdots + c_n \left(\frac{\lambda_2}{\lambda_1}\right)^k x_n \right]$$

    Since $\lambda_1 > \lambda_2$ then fractions $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1}, \ldots < 1$
    and so $\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$ as $k \to \infty$ (for all $i = 2 \ldots n$).

  - **Thus:** $M^k r^{(0)} \approx c_1(\lambda_1^k x_1)$
    - Note if $c_1 = 0$ then the method won’t converge
Random Walk Interpretation

- Imagine a random web surfer:
  - At any time $t$, surfer is on some page $i$
  - At time $t + 1$, the surfer follows an out-link from $i$ uniformly at random
  - Ends up on some page $j$ linked from $i$
  - Process repeats indefinitely

- Let:
  - $p(t)$ ... vector whose $i^{th}$ coordinate is the prob. that the surfer is at page $i$ at time $t$
  - So, $p(t)$ is a probability distribution over pages
The Stationary Distribution

- Where is the surfer at time $t+1$?
  - Follows a link uniformly at random
    \[ p(t+1) = M \cdot p(t) \]
  - Suppose the random walk reaches a state
    \[ p(t+1) = M \cdot p(t) = p(t) \]
    then $p(t)$ is **stationary distribution** of a random walk

- Our original rank vector $r$ satisfies \[ r = M \cdot r \]
  - So, $r$ is a stationary distribution for the random walk
A central result from the theory of random walks (a.k.a. Markov processes):

For graphs that satisfy certain conditions, the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time $t = 0$. 

PageRank: The Google Formulation
PageRank: Three Questions

\[ r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i} \]

or equivalently

\[ r = Mr \]

- Does this converge?
- Does it converge to what we want?
- Are results reasonable?
Does this converge?

- **Example:**

  $r_a = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

  $r_b$

  *Iteration 0, 1, 2, …*
Does it converge to what we want?

Example:

\[ \begin{align*}
  r_a &= 1 \quad 0 \quad 0 \quad 0 \\
  r_b &= 0 \quad 1 \quad 0 \quad 0 \\
\end{align*} \]

Iteration 0, 1, 2, …
PageRank: Problems

2 problems:

- (1) Some pages are **dead ends**
  (have no out-links)
  - Random walk has “nowhere” to go to
  - Such pages cause importance to “leak out”

- (2) **Cycles**:  
  (all out-links are within the group)
  - Random walked gets “stuck” in a trap
  - Eventually cycles absorb all importance
Problem: Cycle

- **Power Iteration:**
  - Set $r_j = 1$
  - $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
  - And iterate

- **Example:**

  \[
  \begin{pmatrix}
  r_y \\
  r_a \\
  r_m
  \end{pmatrix} =
  \begin{pmatrix}
  1/3 & 2/6 & 3/12 & 5/24 & 0 \\
  1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\
  1/3 & 3/6 & 7/12 & 16/24 & 1
  \end{pmatrix}
  \]

  Iteration 0, 1, 2, ...

  All the PageRank score gets “trapped” in node m.
The Google solution for cycles: At each time step, the random surfer has two options
- With prob. $\beta$, follow a link at random
- With prob. $1-\beta$, jump to some random page
- Common values for $\beta$ are in the range 0.8 to 0.9

Surfer will teleport out of cycle within a few time steps
Problem: Dead Ends

- **Power Iteration:**
  - Set \( r_j = 1 \)
  - \( r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i} \)
    - And iterate

- **Example:**

\[
\begin{pmatrix}
    r_y \\
    r_a \\
    r_m
\end{pmatrix} =
\begin{pmatrix}
    1/3 & 2/6 & 3/12 & 5/24 & 0 \\
    1/3 & 1/6 & 2/12 & 3/24 & \vdots & 0 \\
    1/3 & 1/6 & 1/12 & 2/24 & 0
\end{pmatrix}
\]

Iteration 0, 1, 2, ...

Here the PageRank “leaks” out since the matrix is not stochastic.
Solution: Always Teleport!

- **Teleports**: Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly

\[
\begin{array}{ccc}
\ y & \ a & \ m \\
\ y & \frac{1}{2} & \frac{1}{2} & 0 \\
\ a & \frac{1}{2} & 0 & 0 \\
\ m & 0 & \frac{1}{2} & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
\ y & \ a & \ m \\
\ y & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \\
\ a & \frac{1}{2} & 0 & \frac{1}{3} \\
\ m & 0 & \frac{1}{2} & \frac{1}{3} \\
\end{array}
\]
Why are dead-ends and cycles problem and why do teleports solve the problem?

- **Cycles** are not a “problem”, but with them, PageRank scores are **not** what we want
  - **Solution:** Never get stuck in a cycle by teleporting out of it in a finite number of steps

- **Dead-ends** are a problem
  - The matrix is not column stochastic so our initial assumptions are not met
  - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere to go
Solution: Random Teleports

- **Google’s solution that does it all:**
  At each step, random surfer has two options:
  - With probability $\beta$, follow a link at random
  - With probability $1 - \beta$, jump to some random page

- **PageRank equation** [Brin-Page, 98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

This formulation assumes that $M$ has no dead ends. We can either preprocess matrix $M$ to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.
The Google Matrix

- **PageRank equation** [Brin-Page, ‘98]

\[
  r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}
\]

- **The Google Matrix A:**

\[
  A = \beta M + (1 - \beta) \left[ \frac{1}{N} \right]_{N \times N}
\]

- We have a recursive problem: \( r = A \cdot r \)

  And the Power method still works!

- **What is \( \beta \)?**
  - In practice \( \beta = 0.8, 0.9 \) (make 5 steps on avg., jump)
Random Teleports \((\beta = 0.8)\)

\[
M = \begin{pmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 0 \\
0 & \frac{1}{2} & 1
\end{pmatrix}
\]

\[
\left[\frac{1}{N}\right]_{N \times N} = \begin{pmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3}
\end{pmatrix}
\]

\[
A = \begin{pmatrix}
\frac{7}{15} & \frac{7}{15} & \frac{1}{15} \\
\frac{7}{15} & \frac{1}{15} & \frac{1}{15} \\
\frac{1}{15} & \frac{7}{15} & \frac{13}{15}
\end{pmatrix}
\]

\[
y \begin{pmatrix}
\frac{1}{3} \\
\frac{1}{3} \\
\frac{1}{3}
\end{pmatrix}
\begin{pmatrix}
0.33 \\
0.20 \\
0.46
\end{pmatrix}
\begin{pmatrix}
0.24 \\
0.20 \\
0.52
\end{pmatrix}
\begin{pmatrix}
0.26 \\
0.18 \\
0.56
\end{pmatrix}
\begin{pmatrix}
7/33 \\
5/33 \\
21/33
\end{pmatrix}
\]

How do we actually compute the PageRank?
Computing Page Rank

- Key step is matrix-vector multiplication
  - $r_{\text{new}} = A \cdot r_{\text{old}}$
- Easy if we have enough main memory to hold $A$, $r_{\text{old}}$, $r_{\text{new}}$
- Say $N = 1$ billion pages
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix $A$ has $N^2$ entries
    - $10^{18}$ is a large number!

\[
A = \beta \cdot M + (1-\beta) \left[ \frac{1}{N} \right]_{N \times N}
\]

\[
A = \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 0 \\
0 & \frac{1}{2} & 1 \\
\end{bmatrix} + 0.2
\]

\[
= \begin{bmatrix}
7/15 & 7/15 & 1/15 \\
7/15 & 1/15 & 1/15 \\
1/15 & 7/15 & 13/15 \\
\end{bmatrix}
\]
Suppose there are $N$ pages
Consider page $i$, with $d_i$ out-links
We have $M_{ji} = 1/|d_i|$ when $i \rightarrow j$
and $M_{ji} = 0$ otherwise

The random teleport is equivalent to:
- Adding teleport link from $i$ to every other page and setting transition probability to $(1 - \beta)/N$
- Reducing the probability of following each out-link from $1/|d_i|$ to $\beta/|d_i|$ 
- Equivalent: Tax each page a fraction $(1 - \beta)$ of its score and redistribute evenly
Rearranging the Equation

- \( r = A \cdot r, \)  where \( A_{ji} = \beta \ M_{ji} + \frac{1-\beta}{N} \)

- \( r_j = \sum_{i=1}^{N} A_{ji} \cdot r_i \)

- \( r_j = \sum_{i=1}^{N} \left[ \beta \ M_{ji} + \frac{1-\beta}{N} \right] \cdot r_i \)
  
  \[ = \sum_{i=1}^{N} \beta \ M_{ji} \cdot r_i + \frac{1-\beta}{N} \sum_{i=1}^{N} r_i \]

  \[ = \sum_{i=1}^{N} \beta \ M_{ji} \cdot r_i + \frac{1-\beta}{N} \quad \text{since} \ \sum r_i = 1 \]

- **So we get:** \( r = \beta \ M \cdot r + \left[ \frac{1-\beta}{N} \right]_{N} \)

**Note:** Here we assumed \( M \) has no dead-ends

\([x]_N \ldots \text{a vector of length} \ N \text{with all entries} \ x\)
Sparse Matrix Formulation

We just rearranged the **PageRank equation**

\[ r = \beta M \cdot r + \left[ \frac{1 - \beta}{N} \right]_N \]

- where \([ (1-\beta)/N ]_N\) is a vector with all \(N\) entries \((1-\beta)/N\)

- **\(M\) is a sparse matrix!** (with no dead-ends)
  - 10 links per node, approx 10N entries
  - So in each iteration, we need to:
    - Compute \(r^\text{new} = \beta M \cdot r^\text{old}\)
    - Add a constant value \((1-\beta)/N\) to each entry in \(r^\text{new}\)
      - Note if \(M\) contains dead-ends then \(\sum_j r_j^\text{new} < 1\) and we also have to renormalize \(r^\text{new}\) so that it sums to 1
If the graph has no dead-ends then the amount of leaked PageRank is \(1 - \beta\). But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing \(S\).

\[
S = \sum_j r_j^{\text{new}}
\]

**PageRank: The Complete Algorithm**

**Input:** Graph \(G\) and parameter \(\beta\)
- Directed graph \(G\) (can have spider traps and dead ends)
- Parameter \(\beta\)

**Output:** PageRank vector \(r^{\text{new}}\)

1. Set: \(r_j^{\text{old}} = \frac{1}{N}\)
2. Repeat until convergence: 
   \[\sum_j |r_j^{\text{new}} - r_j^{\text{old}}| > \varepsilon\]
   - For all \(j\): 
     \[r_j^{\text{new}} = \sum_{i \to j} \beta \frac{r_i^{\text{old}}}{d_i}\]
     - \(r_j^{\text{new}} = 0\) if in-degree of \(j\) is 0
   - Now re-insert the leaked PageRank:
     \[r_j^{\text{new}} = r_j^{\text{new}} + \frac{1 - S}{N}\]
     where: \(S = \sum_j r_j^{\text{new}}\)

\(r^{\text{old}} = r^{\text{new}}\)
Some Problems with Page Rank

- **Measures generic popularity of a page**
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank (next)

- **Uses a single measure of importance**
  - Other models of importance
  - **Solution:** Hubs-and-Authorities

- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank