MAT 167: Applied Linear Algebra
Lecture 25: Searching by Link Structure II

Naoki Saito

Department of Mathematics
University of California, Davis

June 2 & 5, 2017

Outline

Outline

The Basics of the PageRank algorithm

- The *PageRank* algorithm of Google was invented by Sergey Brin & Larry Page, 1998.
- After webpages retrieved by web crawlers indexed and cataloged, PageRank values are assigned prior to query time according to perceived *importance* so that at query time a ranked list of pages related to the query terms can be presented to the user almost instantaneously.
- PageRank importance is determined by "votes" in the form of links from other pages on the Web. The main ideas:
  1. votes from important sites > votes from less important sites;
  2. the significance of a vote from any source should be tempered (or scaled) by the number of sites the source is voting for (linking to).

The Basics of the PageRank algorithm

- The *PageRank* algorithm of Google was invented by Sergey Brin & Larry Page, 1998.
- After webpages retrieved by web crawlers indexed and cataloged, PageRank values are assigned prior to query time according to perceived *importance* so that at query time a ranked list of pages related to the query terms can be presented to the user almost instantaneously.
- PageRank importance is determined by "votes" in the form of links from other pages on the Web. The main ideas:
  1. votes from important sites $>$ votes from less important sites;
  2. the significance of a vote from any source should be tempered (or scaled) by the number of sites the source is voting for (linking to).

The Basics of the PageRank algorithm

- The *PageRank* algorithm of Google was invented by Sergey Brin & Larry Page, 1998.
- After webpages retrieved by web crawlers indexed and cataloged, PageRank values are assigned prior to query time according to perceived *importance* so that at query time a ranked list of pages related to the query terms can be presented to the user almost instantaneously.
- PageRank importance is determined by "votes" in the form of links from other pages on the Web. The main ideas:
  1. votes from important sites > votes from less important sites;
  2. the significance of a vote from any source should be tempered (or scaled) by the number of sites the source is voting for (linking to).

The Basics of the PageRank algorithm

- The *PageRank* algorithm of Google was invented by Sergey Brin & Larry Page, 1998.
- After webpages retrieved by web crawlers indexed and cataloged, PageRank values are assigned prior to query time according to perceived *importance* so that at query time a ranked list of pages related to the query terms can be presented to the user almost instantaneously.
- PageRank importance is determined by "votes" in the form of links from other pages on the Web. The main ideas:
  1. votes from important sites > votes from less important sites;
  2. the significance of a vote from any source should be tempered (or scaled) by the number of sites the source is voting for (linking to).

The Basics of the PageRank algorithm

- The *PageRank* algorithm of Google was invented by Sergey Brin & Larry Page, 1998.
- After webpages retrieved by web crawlers indexed and cataloged, PageRank values are assigned prior to query time according to perceived *importance* so that at query time a ranked list of pages related to the query terms can be presented to the user almost instantaneously.
- PageRank importance is determined by "votes" in the form of links from other pages on the Web. The main ideas:
  1. votes from important sites $>$ votes from less important sites;
  2. the significance of a vote from any source should be tempered (or scaled) by the number of sites the source is voting for (linking to).

- These ideas can be *recursively* formulated as follows: let $r(P)$ be a rank of a given page P. Then,

$$r(P) = \sum_{Q \in \mathcal{B}_P} \frac{r(Q)}{|Q|},$$

where $\mathcal{B}_P$ is a set of inlink pages to P and $|Q|$ is # of outlinks from Q.

- This needs to be computed iteratively. Suppose there are n pages, $P_1, \ldots, P_n$. First, assign each page an initial ranking, say, $r_0(P_i) = 1/n$ for $i = 1 : n$. Then, successively refine the ranking by computing

$$r_j(P_i) = \sum_{Q \in \mathcal{B}_{P_i}} \frac{r_{j-1}(Q)}{|Q|}, \quad \text{for } j = 1, 2, \ldots$$

- These ideas can be *recursively* formulated as follows: let $r(P)$ be a rank of a given page P. Then,

$$r(P) = \sum_{Q \in \mathcal{B}_P} \frac{r(Q)}{|Q|},$$

  where $\mathcal{B}_P$ is a set of inlink pages to P and $|Q|$ is # of outlinks from Q.
- This needs to be computed iteratively. Suppose there are n pages, $P_1, \ldots, P_n$. First, assign each page an initial ranking, say, $r_0(P_i) = 1/n$ for $i = 1 : n$. Then, successively refine the ranking by computing

$$r_j(P_i) = \sum_{Q \in \mathcal{B}_{P_i}} \frac{r_{j-1}(Q)}{|Q|}, \quad \text{for } j = 1, 2, \ldots$$

- This is accomplished by defining $\boldsymbol{\pi}_j := [r_j(P_1), \cdots, r_j(P_n)]^\mathsf{T} \in \mathbb{R}^n_{\geq 0}$, and iteratively computing:

$$\boldsymbol{\pi}_j^\mathsf{T} = \boldsymbol{\pi}_{j-1}^\mathsf{T} G, \quad \boldsymbol{\pi}_j = \boldsymbol{\pi}_j / \|\boldsymbol{\pi}_j\|_1 \text{ (normalization)} \tag{1}$$

where $G = (g_{ij}) \in \mathbb{R}^{n \times n}_{\geq 0}$ with

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise.} \end{cases}$$

- Again, this is *Power Iteration*! If the limit exists, the *PageRank vector* is defined to be $\boldsymbol{\pi}^\mathsf{T} := \lim_{j \to \infty} \boldsymbol{\pi}_j^\mathsf{T}$, and the ith entry π_i of $\boldsymbol{\pi}$ is the PageRank of P_i. $\boldsymbol{\pi}^\mathsf{T}$ is the dominant *left* eigenvector of G (= the transpose of the dominant eigenvector of G^T).
- For ranking all possible webpages via the PageRank algorithm is certainly a formidable challenge. As Cleve Moler (the founder of MATLAB) pointed out in 2002, this may well be the largest matrix computation ever posed.

- This is accomplished by defining $\boldsymbol{\pi}_j := [r_j(P_1), \cdots, r_j(P_n)]^\mathsf{T} \in \mathbb{R}^n_{\geq 0}$, and iteratively computing:

$$\boldsymbol{\pi}_j^\mathsf{T} = \boldsymbol{\pi}_{j-1}^\mathsf{T} G, \quad \boldsymbol{\pi}_j = \boldsymbol{\pi}_j / \|\boldsymbol{\pi}_j\|_1 \text{ (normalization)} \tag{1}$$

  where $G = (g_{ij}) \in \mathbb{R}^{n \times n}_{\geq 0}$ with

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise.} \end{cases}$$

- Again, this is *Power Iteration*! If the limit exists, the *PageRank vector* is defined to be $\boldsymbol{\pi}^\mathsf{T} := \lim_{j \to \infty} \boldsymbol{\pi}_j^\mathsf{T}$, and the ith entry π_i of $\boldsymbol{\pi}$ is the PageRank of P_i. $\boldsymbol{\pi}^\mathsf{T}$ is the dominant *left* eigenvector of G (= the transpose of the dominant eigenvector of G^T).
- For ranking all possible webpages via the PageRank algorithm is certainly a formidable challenge. As Cleve Moler (the founder of MATLAB) pointed out in 2002, this may well be the largest matrix computation ever posed.

- This is accomplished by defining $\boldsymbol{\pi}_j := [r_j(P_1), \cdots, r_j(P_n)]^\mathsf{T} \in \mathbb{R}^n_{\geq 0}$, and iteratively computing:

$$\boldsymbol{\pi}_j^\mathsf{T} = \boldsymbol{\pi}_{j-1}^\mathsf{T} G, \quad \boldsymbol{\pi}_j = \boldsymbol{\pi}_j / \|\boldsymbol{\pi}_j\|_1 \text{ (normalization)} \tag{1}$$

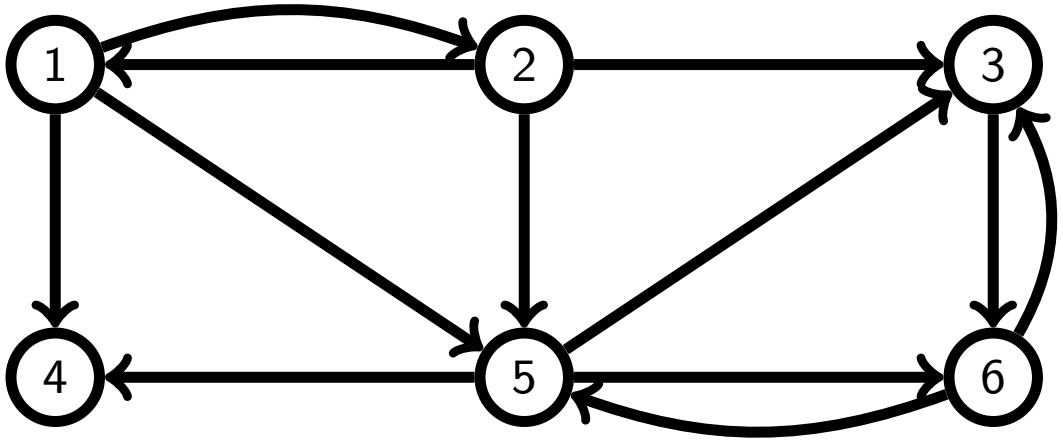
  where $G = (g_{ij}) \in \mathbb{R}^{n \times n}_{\geq 0}$ with

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise.} \end{cases}$$

- Again, this is *Power Iteration*! If the limit exists, the *PageRank vector* is defined to be $\boldsymbol{\pi}^\mathsf{T} := \lim_{j \to \infty} \boldsymbol{\pi}_j^\mathsf{T}$, and the ith entry π_i of $\boldsymbol{\pi}$ is the PageRank of P_i. $\boldsymbol{\pi}^\mathsf{T}$ is the dominant *left* eigenvector of G (= the transpose of the dominant eigenvector of G^T).
- For ranking all possible webpages via the PageRank algorithm is certainly a formidable challenge. As Cleve Moler (the founder of MATLAB) pointed out in 2002, this may well be the largest matrix computation ever posed.

The Familiar Example: Small Web Graph

- Let's consider the same familiar Example 4 of Lecture 2.



- The raw "Google" matrix G constructed by the above algorithm is:

$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

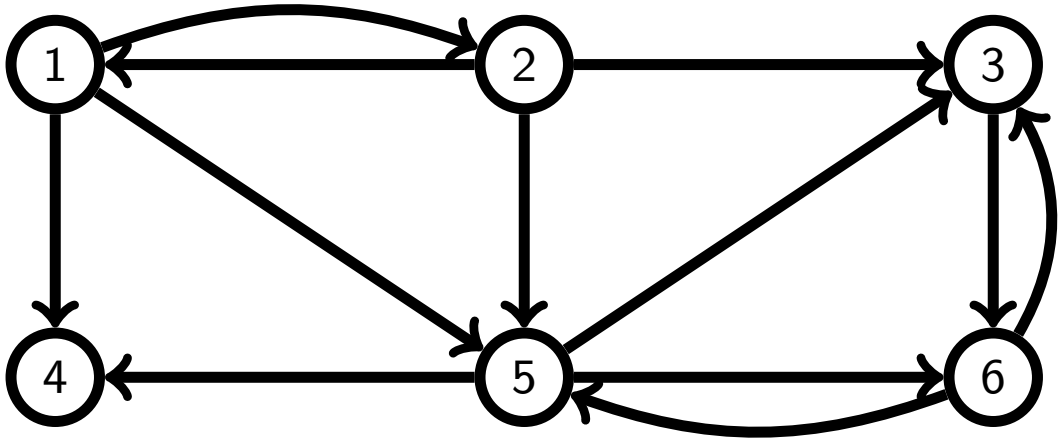
- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^T)$ of G is:

$$\lambda_{\max} = 0.9207; \quad \boldsymbol{\pi}^T = [0.0, 0.0, 0.298335, 0.0793001, 0.219035, 0.403330].$$

- $\lambda_{\max} \neq 1$ is problematic since $\boldsymbol{\pi}^T G = 0.9207\boldsymbol{\pi}^T$, not $\boldsymbol{\pi}^T G = \boldsymbol{\pi}^T$. So, $\boldsymbol{\pi}^T$ is not the limit of Eq. (1) as $j \to \infty$.

The Familiar Example: Small Web Graph

- Let's consider the same familiar Example 4 of Lecture 2.



- The raw "Google" matrix G constructed by the above algorithm is:

$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

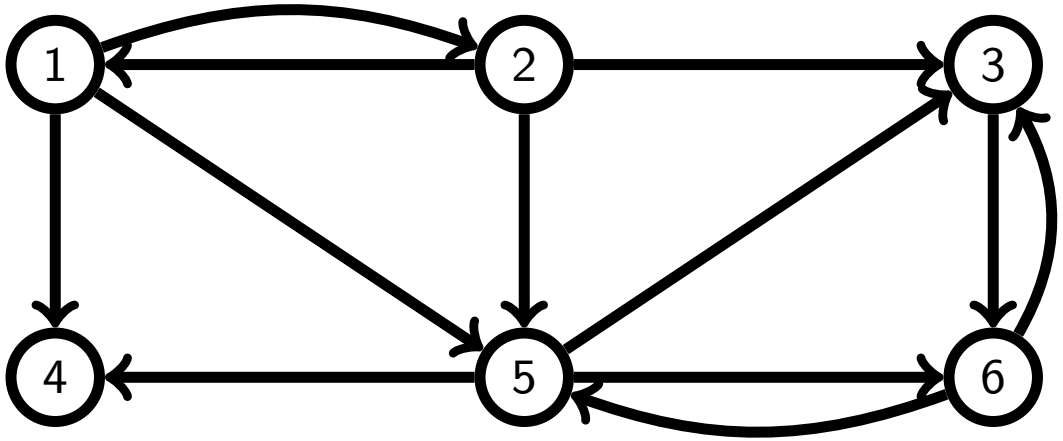
- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^T)$ of G is:

$$\lambda_{\max} = 0.9207; \quad \boldsymbol{\pi}^T = [0.0, 0.0, 0.298335, 0.0793001, 0.219035, 0.403330].$$

- $\lambda_{\max} \neq 1$ is problematic since $\boldsymbol{\pi}^T G = 0.9207\boldsymbol{\pi}^T$, not $\boldsymbol{\pi}^T G = \boldsymbol{\pi}^T$. So, $\boldsymbol{\pi}^T$ is not the limit of Eq. (1) as $j \to \infty$.

The Familiar Example: Small Web Graph

- Let's consider the same familiar Example 4 of Lecture 2.



- The raw "Google" matrix G constructed by the above algorithm is:

$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

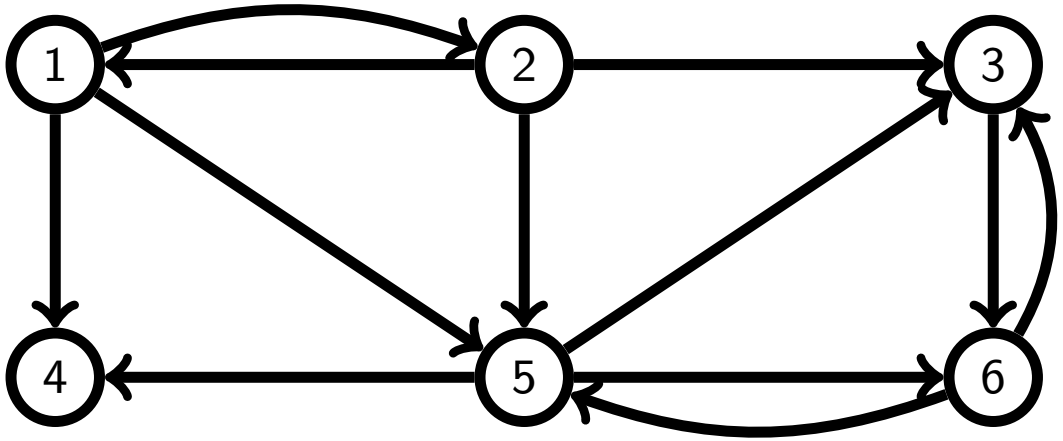
- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^T)$ of G is:

$$\lambda_{\max} = 0.9207; \quad \boldsymbol{\pi}^T = [0.0, 0.0, 0.298335, 0.0793001, 0.219035, 0.403330].$$

- $\lambda_{\max} \neq 1$ is problematic since $\boldsymbol{\pi}^T G = 0.9207\boldsymbol{\pi}^T$, not $\boldsymbol{\pi}^T G = \boldsymbol{\pi}^T$. So, $\boldsymbol{\pi}^T$ is not the limit of Eq. (1) as $j \to \infty$.

The Familiar Example: Small Web Graph

- Let's consider the same familiar Example 4 of Lecture 2.



- The raw "Google" matrix G constructed by the above algorithm is:

$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^T)$ of G is:

$$\lambda_{\max} = 0.9207; \quad \boldsymbol{\pi}^T = [0.0, 0.0, 0.298335, 0.0793001, 0.219035, 0.403330].$$

- $\lambda_{\max} \neq 1$ is problematic since $\boldsymbol{\pi}^T G = 0.9207\boldsymbol{\pi}^T$, not $\boldsymbol{\pi}^T G = \boldsymbol{\pi}^T$. So, $\boldsymbol{\pi}^T$ is not the limit of Eq. (1) as $j \to \infty$.

Outline

Markov Model of the Web

- The "raw" Google matrix $G = (g_{ij}) \in \mathbb{R}^{n \times n}_{\geq 0}$ has $g_{ij} \geq 0$ for all i, j, and its *row sums*, i.e., $\sum_{j=1}^{n} g_{ij}$ is either 1 or 0.
- Zero row sums correspond to nodes (i.e., pages) that have no outlinks. Such nodes/pages are referred to as *dangling* nodes. In our familiar example of the 6 node web graph, Node 4 is a dangling node.
- Let's assume for a moment that there are no dangling nodes in a given web graph, or if there are such nodes, they are accounted for by artificially adding appropriate links to make all the row sums equal 1. E.g., adding artificial outlinks from Node 4 to all the nodes including Node 4 itself amounts to replacing the 4th row of G by $[1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$.
- If each row sum of a nonnegative matrix G equals 1, such a matrix is called a *row stochastic matrix*.
- Then, the recursion Eq. (1): $\boldsymbol{\pi}_j^{\mathsf{T}} = \boldsymbol{\pi}_{j-1}^{\mathsf{T}} G$ represents the evolution of a *Markov chain*, or more precisely, a *random walk* on the graph defined by the link structure of the webpages in Google's database.

Markov Model of the Web

- The "raw" Google matrix $G = (g_{ij}) \in \mathbb{R}_{\geq 0}^{n \times n}$ has $g_{ij} \geq 0$ for all i, j, and its *row sums*, i.e., $\sum_{j=1}^{n} g_{ij}$ is either 1 or 0.
- Zero row sums correspond to nodes (i.e., pages) that have no outlinks. Such nodes/pages are referred to as *dangling* nodes. In our familiar example of the 6 node web graph, Node 4 is a dangling node.
- Let's assume for a moment that there are no dangling nodes in a given web graph, or if there are such nodes, they are accounted for by artificially adding appropriate links to make all the row sums equal 1. E.g., adding artificial outlinks from Node 4 to all the nodes including Node 4 itself amounts to replacing the 4th row of G by $[1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$.
- If each row sum of a nonnegative matrix G equals 1, such a matrix is called a *row stochastic matrix*.
- Then, the recursion Eq. (1): $\boldsymbol{\pi}_j^{\mathsf{T}} = \boldsymbol{\pi}_{j-1}^{\mathsf{T}} G$ represents the evolution of a *Markov chain*, or more precisely, a *random walk* on the graph defined by the link structure of the webpages in Google's database.

Markov Model of the Web

- The "raw" Google matrix $G = (g_{ij}) \in \mathbb{R}_{\geq 0}^{n \times n}$ has $g_{ij} \geq 0$ for all i, j, and its *row sums*, i.e., $\sum_{j=1}^{n} g_{ij}$ is either 1 or 0.
- Zero row sums correspond to nodes (i.e., pages) that have no outlinks. Such nodes/pages are referred to as *dangling* nodes. In our familiar example of the 6 node web graph, Node 4 is a dangling node.
- Let's assume for a moment that there are no dangling nodes in a given web graph, or if there are such nodes, they are accounted for by artificially adding appropriate links to make all the row sums equal 1. E.g., adding artificial outlinks from Node 4 to all the nodes including Node 4 itself amounts to replacing the 4th row of G by $[1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$.
- If each row sum of a nonnegative matrix G equals 1, such a matrix is called a *row stochastic matrix*.
- Then, the recursion Eq. (1): $\boldsymbol{\pi}_j^{\mathsf{T}} = \boldsymbol{\pi}_{j-1}^{\mathsf{T}} G$ represents the evolution of a *Markov chain*, or more precisely, a *random walk* on the graph defined by the link structure of the webpages in Google's database.

Markov Model of the Web

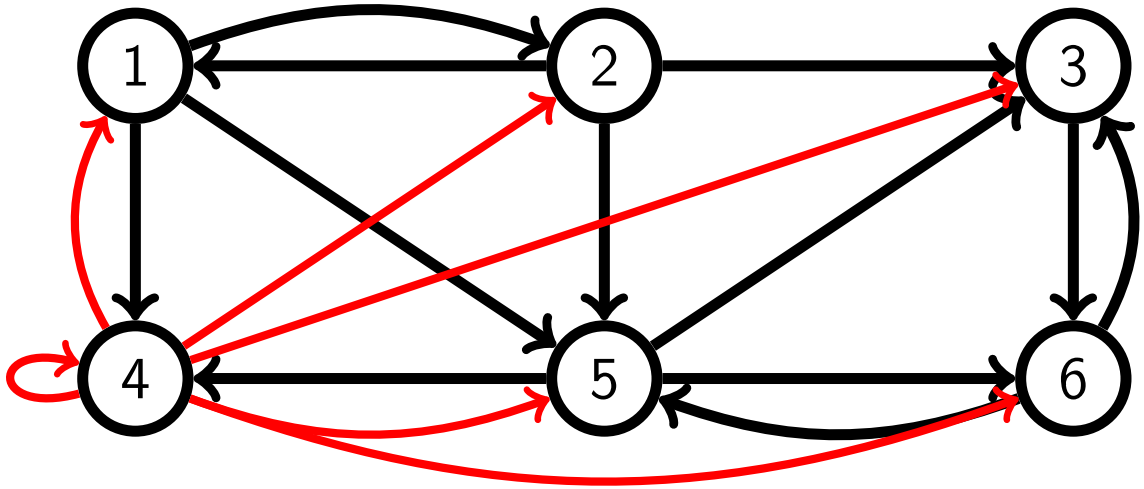
- The "raw" Google matrix $G = (g_{ij}) \in \mathbb{R}^{n \times n}_{\geq 0}$ has $g_{ij} \geq 0$ for all i, j, and its *row sums*, i.e., $\sum_{j=1}^{n} g_{ij}$ is either 1 or 0.
- Zero row sums correspond to nodes (i.e., pages) that have no outlinks. Such nodes/pages are referred to as *dangling* nodes. In our familiar example of the 6 node web graph, Node 4 is a dangling node.
- Let's assume for a moment that there are no dangling nodes in a given web graph, or if there are such nodes, they are accounted for by artificially adding appropriate links to make all the row sums equal 1. E.g., adding artificial outlinks from Node 4 to all the nodes including Node 4 itself amounts to replacing the 4th row of G by $[1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$.
- If each row sum of a nonnegative matrix G equals 1, such a matrix is called a *row stochastic matrix*.
- Then, the recursion Eq. (1): $\boldsymbol{\pi}_j^{\mathsf{T}} = \boldsymbol{\pi}_{j-1}^{\mathsf{T}} G$ represents the evolution of a *Markov chain*, or more precisely, a *random walk* on the graph defined by the link structure of the webpages in Google's database.

Markov Model of the Web

- The "raw" Google matrix $G = (g_{ij}) \in \mathbb{R}_{\geq 0}^{n \times n}$ has $g_{ij} \geq 0$ for all i, j, and its *row sums*, i.e., $\sum_{j=1}^{n} g_{ij}$ is either 1 or 0.
- Zero row sums correspond to nodes (i.e., pages) that have no outlinks. Such nodes/pages are referred to as *dangling* nodes. In our familiar example of the 6 node web graph, Node 4 is a dangling node.
- Let's assume for a moment that there are no dangling nodes in a given web graph, or if there are such nodes, they are accounted for by artificially adding appropriate links to make all the row sums equal 1. E.g., adding artificial outlinks from Node 4 to all the nodes including Node 4 itself amounts to replacing the 4th row of G by $[1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$.
- If each row sum of a nonnegative matrix G equals 1, such a matrix is called a *row stochastic matrix*.
- Then, the recursion Eq. (1): $\boldsymbol{\pi}_j^{\mathsf{T}} = \boldsymbol{\pi}_{j-1}^{\mathsf{T}} G$ represents the evolution of a *Markov chain*, or more precisely, a *random walk* on the graph defined by the link structure of the webpages in Google's database.

The Familiar Example: Small Web Graph

- Let's add artificial outlinks from Node 4:



- The adjusted "Google" matrix G, which is now *row stochastic*, is:

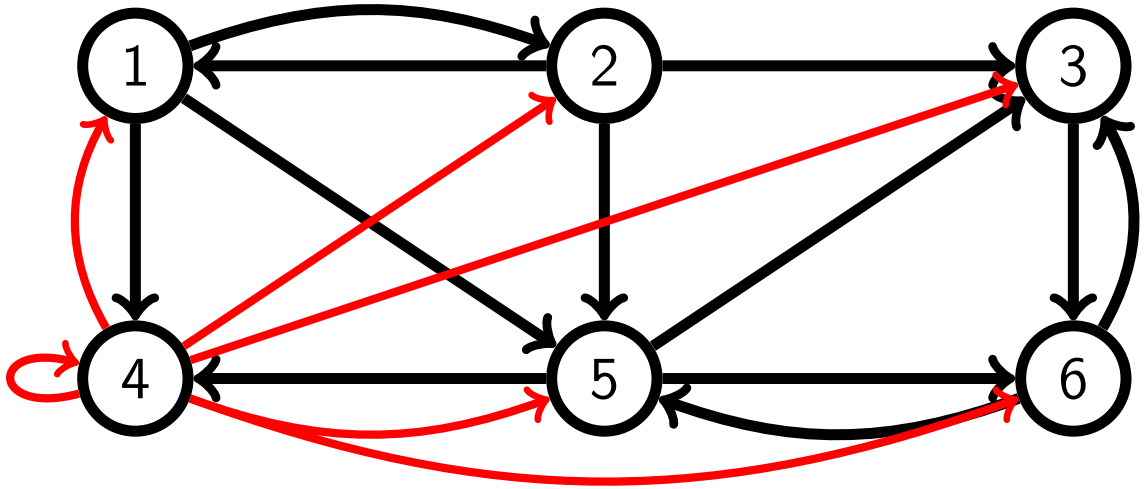
$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^{\mathsf{T}})$ of G is:

$$\lambda_{\max} = 1; \quad \boldsymbol{\pi}^{\mathsf{T}} = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

The Familiar Example: Small Web Graph

- Let's add artificial outlinks from Node 4:



- The adjusted "Google" matrix G, which is now *row stochastic*, is:

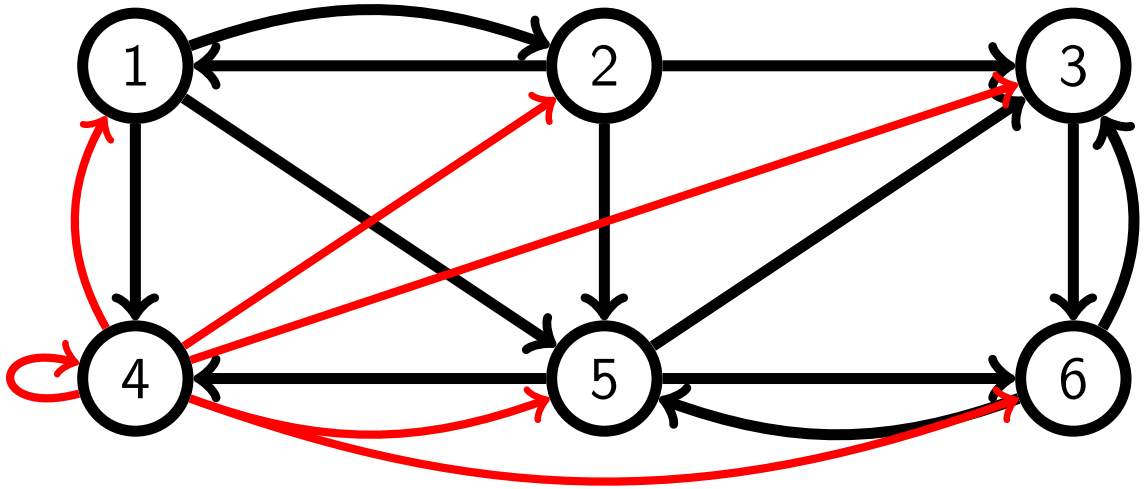
$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^\top)$ of G is:

$$\lambda_{\max} = 1; \quad \boldsymbol{\pi}^\top = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

The Familiar Example: Small Web Graph

- Let's add artificial outlinks from Node 4:



- The adjusted "Google" matrix G, which is now *row stochastic*, is:

$$G = \begin{bmatrix} 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

- The dominant left eigenpair $(\lambda_{\max}, \boldsymbol{\pi}^{\mathsf{T}})$ of G is:

$$\lambda_{\max} = 1; \quad \boldsymbol{\pi}^{\mathsf{T}} = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3
- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3

- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3
- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3

- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3
- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3
- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

The Familiar Example: Small Web Graph ...

- These PageRank values are query-independent.
- Now suppose a query is entered containing term 1 and term 2, and suppose the inverted term-document file looks like:

  term 1: doc 1, doc 4, doc 6

  term 2: doc 1, doc 3
- Hence, the relevancy set for a query on terms 1 and 2 is $\{1,3,4,6\}$
- Then, check their PageRank values $\pi_1, \pi_3, \pi_4, \pi_6$ and sort them in the nonincreasing order to get:

$$\pi_6 = 0.365079$$
$$\pi_3 = 0.277778$$
$$\pi_4 = 0.0952381$$
$$\pi_1 = 0.0238095$$

- Consequently, document 6 is the most important among the relevant documents followed by documents 3, 4, and 1.

Further Probabilistic Interpretations

- Instead of using the original probability distribution assignment:

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise,} \end{cases}$$

  other suitable probability distribution may be used across the rows.
- For example, in the Familiar Example, if the Web usage logs show that users accessing Page/Node 6 are twice as likely to jump to Page/Node 5 as they are to jump to Page/Node 3, then the 6th row of G should be replaced by $[0, 0, 1/3, 0, 2/3, 0]$.
- The resulting modified G is still row stochastic of course, and the dominant left eigenvector now becomes:

$$\boldsymbol{\pi}^\mathsf{T} = [0.0291262, 0.0291262, 0.228155, 0.116505, 0.262136, 0.334951].$$

  Compare this with the homogeneous probability distribution case:

$$\boldsymbol{\pi}^\mathsf{T} = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

- Clearly, PageRank value of Node 5 $\uparrow$ while that of Node 3 $\downarrow$.

Further Probabilistic Interpretations

- Instead of using the original probability distribution assignment:

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise,} \end{cases}$$

  other suitable probability distribution may be used across the rows.
- For example, in the Familiar Example, if the Web usage logs show that users accessing Page/Node 6 are twice as likely to jump to Page/Node 5 as they are to jump to Page/Node 3, then the 6th row of G should be replaced by $[0, 0, 1/3, 0, 2/3, 0]$.
- The resulting modified G is still row stochastic of course, and the dominant left eigenvector now becomes:

$$\boldsymbol{\pi}^\top = [0.0291262, 0.0291262, 0.228155, 0.116505, 0.262136, 0.334951].$$

  Compare this with the homogeneous probability distribution case:

$$\boldsymbol{\pi}^\top = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

- Clearly, PageRank value of Node 5 $\uparrow$ while that of Node 3 $\downarrow$.

Further Probabilistic Interpretations

- Instead of using the original probability distribution assignment:

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise,} \end{cases}$$

  other suitable probability distribution may be used across the rows.
- For example, in the Familiar Example, if the Web usage logs show that users accessing Page/Node 6 are twice as likely to jump to Page/Node 5 as they are to jump to Page/Node 3, then the 6th row of G should be replaced by $[0, 0, 1/3, 0, 2/3, 0]$.
- The resulting modified G is still row stochastic of course, and the dominant left eigenvector now becomes:

$$\boldsymbol{\pi}^\mathsf{T} = [0.0291262, 0.0291262, 0.228155, 0.116505, 0.262136, 0.334951].$$

  Compare this with the homogeneous probability distribution case:

$$\boldsymbol{\pi}^\mathsf{T} = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

- Clearly, PageRank value of Node 5 $\uparrow$ while that of Node 3 $\downarrow$.

Further Probabilistic Interpretations

- Instead of using the original probability distribution assignment:

$$g_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j; \\ 0 & \text{otherwise,} \end{cases}$$

  other suitable probability distribution may be used across the rows.
- For example, in the Familiar Example, if the Web usage logs show that users accessing Page/Node 6 are twice as likely to jump to Page/Node 5 as they are to jump to Page/Node 3, then the 6th row of G should be replaced by $[0, 0, 1/3, 0, 2/3, 0]$.
- The resulting modified G is still row stochastic of course, and the dominant left eigenvector now becomes:

$$\boldsymbol{\pi}^{\mathsf{T}} = [0.0291262, 0.0291262, 0.228155, 0.116505, 0.262136, 0.334951].$$

  Compare this with the homogeneous probability distribution case:

$$\boldsymbol{\pi}^{\mathsf{T}} = [0.0238095, 0.0238095, 0.277778, 0.0952381, 0.214286, 0.365079]$$

- Clearly, PageRank value of Node 5 $\uparrow$ while that of Node 3 $\downarrow$.

Further Probabilistic Interpretations ...

- In general, the dominant eigenvalue for every stochastic matrix is 1.
- Hence, if the PageRank iteration Eq. (1) converges, it converges to the normalized left eigenvector $\boldsymbol{\pi}^\top$ satisfying:

$$\boldsymbol{\pi}^\top = \boldsymbol{\pi}^\top G, \quad \boldsymbol{\pi}^\top \mathbf{1}_n = 1,$$

  where $\mathbf{1}_n = [1, 1, \cdots, 1]^\top \in \mathbb{R}^n$.
- $\boldsymbol{\pi}^\top$ represents the stationary (or steady-state) distribution of the Markov chain/random walk.
- Hence, Google intuitively characterizes *the PageRank value of each site as the long-run proportion of time spent at that site by a Web surfer eternally clicking on links at random* (clicking back or entering a URL is excluded in this model).

Further Probabilistic Interpretations ...

- In general, the dominant eigenvalue for every stochastic matrix is 1.
- Hence, if the PageRank iteration Eq. (1) converges, it converges to the normalized left eigenvector $\boldsymbol{\pi}^\mathsf{T}$ satisfying:

$$\boldsymbol{\pi}^\mathsf{T} = \boldsymbol{\pi}^\mathsf{T} G, \quad \boldsymbol{\pi}^\mathsf{T} \mathbf{1}_n = 1,$$

where $\mathbf{1}_n = [1, 1, \cdots, 1]^\mathsf{T} \in \mathbb{R}^n$.

- $\boldsymbol{\pi}^\mathsf{T}$ represents the stationary (or steady-state) distribution of the Markov chain/random walk.
- Hence, Google intuitively characterizes *the PageRank value of each site as the long-run proportion of time spent at that site by a Web surfer eternally clicking on links at random* (clicking back or entering a URL is excluded in this model).

Further Probabilistic Interpretations ...

- In general, the dominant eigenvalue for every stochastic matrix is 1.
- Hence, if the PageRank iteration Eq. (1) converges, it converges to the normalized left eigenvector $\boldsymbol{\pi}^\mathsf{T}$ satisfying:

$$\boldsymbol{\pi}^\mathsf{T} = \boldsymbol{\pi}^\mathsf{T} G, \quad \boldsymbol{\pi}^\mathsf{T} \mathbf{1}_n = 1,$$

  where $\mathbf{1}_n = [1, 1, \cdots, 1]^\mathsf{T} \in \mathbb{R}^n$.
- $\boldsymbol{\pi}^\mathsf{T}$ represents the stationary (or steady-state) distribution of the Markov chain/random walk.
- Hence, Google intuitively characterizes *the PageRank value of each site as the long-run proportion of time spent at that site by a Web surfer eternally clicking on links at random* (clicking back or entering a URL is excluded in this model).

Further Probabilistic Interpretations ...

- In general, the dominant eigenvalue for every stochastic matrix is 1.
- Hence, if the PageRank iteration Eq. (1) converges, it converges to the normalized left eigenvector $\boldsymbol{\pi}^\mathsf{T}$ satisfying:

$$\boldsymbol{\pi}^\mathsf{T} = \boldsymbol{\pi}^\mathsf{T} G, \quad \boldsymbol{\pi}^\mathsf{T} \mathbf{1}_n = 1,$$

  where $\mathbf{1}_n = [1, 1, \cdots, 1]^\mathsf{T} \in \mathbb{R}^n$.
- $\boldsymbol{\pi}^\mathsf{T}$ represents the stationary (or steady-state) distribution of the Markov chain/random walk.
- Hence, Google intuitively characterizes *the PageRank value of each site as the long-run proportion of time spent at that site by a Web surfer eternally clicking on links at random* (clicking back or entering a URL is excluded in this model).

PageRank Adjustments

- Other than artificially adding outlinks to make G row stochastic, another greater difficulty can arise.
- An *irreducible* Markov chain is one where every node can be ultimately reached from all other nodes. In other words, there is a *path* from Node i to Node j for all i, j.
- In addition, one can show that the stationary distribution π^T for such a Markov chain is unique and positive (thanks to the so-called *Perron-Frobenius Theorem*), which are desirable properties for a PageRank vector.
- However, in reality, the raw Google matrix G even after making it row stochastic, the underlying Markov chain may be most likely *reducible*, i.e., there exists a subset of nodes in the graph in which a random walk eventually becomes trapped, and cannot escape to the outside of that group of nodes.

PageRank Adjustments

- Other than artificially adding outlinks to make G row stochastic, another greater difficulty can arise.
- An *irreducible* Markov chain is one where every node can be ultimately reached from all other nodes. In other words, there is a *path* from Node i to Node j for all i, j.
- In addition, one can show that the stationary distribution $\pi^\top$ for such a Markov chain is unique and positive (thanks to the so-called *Perron-Frobenius Theorem*), which are desirable properties for a PageRank vector.
- However, in reality, the raw Google matrix G even after making it row stochastic, the underlying Markov chain may be most likely *reducible*, i.e., there exists a subset of nodes in the graph in which a random walk eventually becomes trapped, and cannot escape to the outside of that group of nodes.

PageRank Adjustments

- Other than artificially adding outlinks to make G row stochastic, another greater difficulty can arise.
- An *irreducible* Markov chain is one where every node can be ultimately reached from all other nodes. In other words, there is a *path* from Node i to Node j for all i, j.
- In addition, one can show that the stationary distribution $\boldsymbol{\pi}^{\mathsf{T}}$ for such a Markov chain is unique and positive (thanks to the so-called *Perron-Frobenius Theorem*), which are desirable properties for a PageRank vector.
- However, in reality, the raw Google matrix G even after making it row stochastic, the underlying Markov chain may be most likely *reducible*, i.e., there exists a subset of nodes in the graph in which a random walk eventually becomes trapped, and cannot escape to the outside of that group of nodes.

PageRank Adjustments

- Other than artificially adding outlinks to make G row stochastic, another greater difficulty can arise.
- An *irreducible* Markov chain is one where every node can be ultimately reached from all other nodes. In other words, there is a *path* from Node i to Node j for all i, j.
- In addition, one can show that the stationary distribution $\boldsymbol{\pi}^T$ for such a Markov chain is unique and positive (thanks to the so-called *Perron-Frobenius Theorem*), which are desirable properties for a PageRank vector.
- However, in reality, the raw Google matrix G even after making it row stochastic, the underlying Markov chain may be most likely *reducible*, i.e., there exists a subset of nodes in the graph in which a random walk eventually becomes trapped, and cannot escape to the outside of that group of nodes.

PageRank Adjustments ...

- Brin and Page force irreducibility into the picture by making every node directly reachable from every other node by *perturbing* G as

$$\widetilde{G} = \alpha G + (1-\alpha)E \quad \text{where } 0 < \alpha < 1 \text{ and } E := \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^{\mathsf{T}},$$

  which is a *convex* combination of two stochastic matrices; hence $\widetilde{G}$ is stochastic and irreducible, i.e., has a unique stationary distribution $\boldsymbol{\pi}^{\mathsf{T}}$.

- The Google reasoning: this stochastic matrix $\widetilde{G}$ models a Web surfer's "teleportation" tendency to randomly jump to a new page by entering a URL on the address line, and it assumes that each URL has an equal likelihood of being selected.

- There are many more ways to force irreducibility of the Markov chain/random walk. For the details, see Reference 1: Langville and Meyer (2005).

PageRank Adjustments ...

- Brin and Page force irreducibility into the picture by making every node directly reachable from every other node by *perturbing* G as

$$\widetilde{G} = \alpha G + (1-\alpha)E \quad \text{where } 0 < \alpha < 1 \text{ and } E := \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^{\mathsf{T}},$$

  which is a *convex* combination of two stochastic matrices; hence $\widetilde{G}$ is stochastic and irreducible, i.e., has a unique stationary distribution $\boldsymbol{\pi}^{\mathsf{T}}$.
- The Google reasoning: this stochastic matrix $\widetilde{G}$ models a Web surfer's "teleportation" tendency to randomly jump to a new page by entering a URL on the address line, and it assumes that each URL has an equal likelihood of being selected.
- There are many more ways to force irreducibility of the Markov chain/random walk. For the details, see Reference 1: Langville and Meyer (2005).

PageRank Adjustments ...

- Brin and Page force irreducibility into the picture by making every node directly reachable from every other node by *perturbing* G as

$$\widetilde{G} = \alpha G + (1-\alpha)E \quad \text{where } 0 < \alpha < 1 \text{ and } E := \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^{\mathsf{T}},$$

  which is a *convex* combination of two stochastic matrices; hence $\widetilde{G}$ is stochastic and irreducible, i.e., has a unique stationary distribution $\boldsymbol{\pi}^{\mathsf{T}}$.
- The Google reasoning: this stochastic matrix $\widetilde{G}$ models a Web surfer's "teleportation" tendency to randomly jump to a new page by entering a URL on the address line, and it assumes that each URL has an equal likelihood of being selected.
- There are many more ways to force irreducibility of the Markov chain/random walk. For the details, see Reference 1: Langville and Meyer (2005).

PageRank Adjustments ...

- Later, Google adopted a more realistic and less democratic stance by using a better and more flexible perturbation matrix,

$$E := \mathbf{1}_n \boldsymbol{v}^\mathsf{T},$$

  where the "personalization" vector $\boldsymbol{v}^\mathsf{T} > \mathbf{0}_n^\mathsf{T}$ is a probability vector that allows non-uniform probabilities for teleporting to particular pages.

- More importantly, at least from a business point of view, taking the perturbation to be of the form $E = \mathbf{1}_n \boldsymbol{v}^\mathsf{T}$ permits "intervention" by fiddling with $\boldsymbol{v}^\mathsf{T}$ to adjust PageRank values up or down according to commercial considerations!

PageRank Adjustments ...

- Later, Google adopted a more realistic and less democratic stance by using a better and more flexible perturbation matrix,

$$E := \mathbf{1}_n \boldsymbol{v}^\mathsf{T},$$

  where the "personalization" vector $\boldsymbol{v}^\mathsf{T} > \mathbf{0}_n^\mathsf{T}$ is a probability vector that allows non-uniform probabilities for teleporting to particular pages.
- More importantly, at least from a business point of view, taking the perturbation to be of the form $E = \mathbf{1}_n \boldsymbol{v}^\mathsf{T}$ permits "intervention" by fiddling with $\boldsymbol{v}^\mathsf{T}$ to adjust PageRank values up or down according to commercial considerations!

Outline

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation

- PageRank does not give a relevancy score for given query terms; it just gives an importance score for each webpage.
- Google combines PageRank and other scores to give an overall ranking for given query terms.
- A basic Web information retrieval model using PageRank is the following:
  1. A full web document scan determines the subset of nodes containing the query terms, which is called the relevancy set for the query.
  2. The relevancy set is sorted according to the PageRank scores of each document in the set.
- It is reported that Google updates PageRank once every few weeks for all documents in Web collection at some point in the past. However, it is not entirely clear how often they update the PageRank scores of the Web collection documents.
- Since computing $\boldsymbol{\pi}^{\mathsf{T}}$ is expensive and the Web's structure is changing all the time, how to update the PageRank vector is an active area of research!

PageRank Implementation ...

- The power iteration is applied to $\boldsymbol{\pi}_{j+1}^{\mathsf{T}} = \alpha \boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G} + (1-\alpha)\boldsymbol{v}^{\mathsf{T}}$ with $\boldsymbol{\pi}_0^{\mathsf{T}} = \mathbf{1}_n^{\mathsf{T}}/n$.
- Although n is huge, this iterative computation has some advantages:
  1. it does not destroy the inherent extreme *sparsity* of $\widetilde{G}$;
  2. $\boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G}$ requires only *sparse inner products*, which are in turn easily implemented in *parallel*.
- Brin and Page claimed that useful results are obtained after 50 iterations for the case $n = 322{,}000{,}000$.
- Several other iterative methods other than the simple power iteration have been proposed.

PageRank Implementation ...

- The power iteration is applied to $\boldsymbol{\pi}_{j+1}^{\mathsf{T}} = \alpha \boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G} + (1-\alpha)\boldsymbol{v}^{\mathsf{T}}$ with $\boldsymbol{\pi}_0^{\mathsf{T}} = \mathbf{1}_n^{\mathsf{T}}/n$.
- Although n is huge, this iterative computation has some advantages:
  1. it does not destroy the inherent extreme *sparsity* of $\widetilde{G}$;
  2. $\boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G}$ requires only *sparse inner products*, which are in turn easily implemented in *parallel*.
- Brin and Page claimed that useful results are obtained after 50 iterations for the case $n = 322{,}000{,}000$.
- Several other iterative methods other than the simple power iteration have been proposed.

PageRank Implementation ...

- The power iteration is applied to $\boldsymbol{\pi}_{j+1}^{\mathsf{T}} = \alpha \boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G} + (1-\alpha)\boldsymbol{v}^{\mathsf{T}}$ with $\boldsymbol{\pi}_0^{\mathsf{T}} = \mathbf{1}_n^{\mathsf{T}}/n$.
- Although n is huge, this iterative computation has some advantages:
  1. it does not destroy the inherent extreme *sparsity* of $\widetilde{G}$;
  2. $\boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G}$ requires only *sparse inner products*, which are in turn easily implemented in *parallel*.
- Brin and Page claimed that useful results are obtained after 50 iterations for the case $n = 322{,}000{,}000$.
- Several other iterative methods other than the simple power iteration have been proposed.

PageRank Implementation ...

- The power iteration is applied to $\boldsymbol{\pi}_{j+1}^{\mathsf{T}} = \alpha \boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G} + (1-\alpha)\boldsymbol{v}^{\mathsf{T}}$ with $\boldsymbol{\pi}_0^{\mathsf{T}} = \mathbf{1}_n^{\mathsf{T}}/n$.
- Although n is huge, this iterative computation has some advantages:
  1. it does not destroy the inherent extreme *sparsity* of $\widetilde{G}$;
  2. $\boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G}$ requires only *sparse inner products*, which are in turn easily implemented in *parallel*.
- Brin and Page claimed that useful results are obtained after 50 iterations for the case $n = 322{,}000{,}000$.
- Several other iterative methods other than the simple power iteration have been proposed.

PageRank Implementation ...

- The power iteration is applied to $\boldsymbol{\pi}_{j+1}^{\mathsf{T}} = \alpha \boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G} + (1-\alpha)\boldsymbol{v}^{\mathsf{T}}$ with $\boldsymbol{\pi}_0^{\mathsf{T}} = \mathbf{1}_n^{\mathsf{T}}/n$.
- Although n is huge, this iterative computation has some advantages:
  1. it does not destroy the inherent extreme *sparsity* of $\widetilde{G}$;
  2. $\boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G}$ requires only *sparse inner products*, which are in turn easily implemented in *parallel*.
- Brin and Page claimed that useful results are obtained after 50 iterations for the case $n = 322{,}000{,}000$.
- Several other iterative methods other than the simple power iteration have been proposed.

PageRank Implementation ...

- The power iteration is applied to $\boldsymbol{\pi}_{j+1}^{\mathsf{T}} = \alpha \boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G} + (1-\alpha)\boldsymbol{v}^{\mathsf{T}}$ with $\boldsymbol{\pi}_0^{\mathsf{T}} = \mathbf{1}_n^{\mathsf{T}}/n$.
- Although n is huge, this iterative computation has some advantages:
  1. it does not destroy the inherent extreme *sparsity* of $\widetilde{G}$;
  2. $\boldsymbol{\pi}_j^{\mathsf{T}} \widetilde{G}$ requires only *sparse inner products*, which are in turn easily implemented in *parallel*.
- Brin and Page claimed that useful results are obtained after 50 iterations for the case $n = 322{,}000{,}000$.
- Several other iterative methods other than the simple power iteration have been proposed.

Weaknesses of PageRank

- Topic drift: similarly to the HITS case, it is possible that a very authoritative yet off-topic document be linked to a document containing the query terms. This very authoritative document can carry so much weight that it and its neighboring documents dominate the relevant ranked list returned to the user, skewing the results towards off-topic documents.
- Much work, thought, and heuristics must be applied by Google engineers to determine the relevancy score; otherwise, no matter how good PageRank is, the ranked list returned to the user is of little value if the pages are off-topic.
- Does *importance* of a webpage really serve as a good proxy to *relevance*? PageRank itself cannot distinguish between pages that are generally authoritative and pages that are authoritative on the query topic.

Weaknesses of PageRank

- Topic drift: similarly to the HITS case, it is possible that a very authoritative yet off-topic document be linked to a document containing the query terms. This very authoritative document can carry so much weight that it and its neighboring documents dominate the relevant ranked list returned to the user, skewing the results towards off-topic documents.
- Much work, thought, and heuristics must be applied by Google engineers to determine the relevancy score; otherwise, no matter how good PageRank is, the ranked list returned to the user is of little value if the pages are off-topic.
- Does *importance* of a webpage really serve as a good proxy to *relevance*? PageRank itself cannot distinguish between pages that are generally authoritative and pages that are authoritative on the query topic.

Weaknesses of PageRank

- Topic drift: similarly to the HITS case, it is possible that a very authoritative yet off-topic document be linked to a document containing the query terms. This very authoritative document can carry so much weight that it and its neighboring documents dominate the relevant ranked list returned to the user, skewing the results towards off-topic documents.
- Much work, thought, and heuristics must be applied by Google engineers to determine the relevancy score; otherwise, no matter how good PageRank is, the ranked list returned to the user is of little value if the pages are off-topic.
- Does *importance* of a webpage really serve as a good proxy to *relevance*? PageRank itself cannot distinguish between pages that are generally authoritative and pages that are authoritative on the query topic.

Strengths of PageRank

+ On the other hand, the use of importance, rather than relevance, is the key to Google's success. PageRank is query independent, which is a major advantage over HITS's query dependence.
+ Much more spam-resistant than HITS; it's very hard for a webpage owner to add inlinks into his page from other important pages. Even if he/she succeeds in doing this, the increase of its PageRank value will likely be inconsequential since PageRank is a global measure compared to the local nature of HITS.
+ Flexibility of the "personalization" vector $\boldsymbol{v}^T$ that Google is free to choose in defining the fudge-factor term $E = \mathbf{1}_n \boldsymbol{v}^T$, which may defeat some users who try to manipulate PageRank.

Strengths of PageRank

+ On the other hand, the use of importance, rather than relevance, is the key to Google's success. PageRank is query independent, which is a major advantage over HITS's query dependence.
+ Much more spam-resistant than HITS; it's very hard for a webpage owner to add inlinks into his page from other important pages. Even if he/she succeeds in doing this, the increase of its PageRank value will likely be inconsequential since PageRank is a global measure compared to the local nature of HITS.
+ Flexibility of the "personalization" vector $\boldsymbol{v}^T$ that Google is free to choose in defining the fudge-factor term $E = \mathbf{1}_n \boldsymbol{v}^T$, which may defeat some users who try to manipulate PageRank.

Strengths of PageRank

+ On the other hand, the use of importance, rather than relevance, is the key to Google's success. PageRank is query independent, which is a major advantage over HITS's query dependence.
+ Much more spam-resistant than HITS; it's very hard for a webpage owner to add inlinks into his page from other important pages. Even if he/she succeeds in doing this, the increase of its PageRank value will likely be inconsequential since PageRank is a global measure compared to the local nature of HITS.
+ Flexibility of the "personalization" vector $\boldsymbol{v}^{\mathsf{T}}$ that Google is free to choose in defining the fudge-factor term $E = \mathbf{1}_n \boldsymbol{v}^{\mathsf{T}}$, which may defeat some users who try to manipulate PageRank.

References

1. A. N. Langville and C. D. Meyer: "A survey of eigenvector methods for Web information retrieval," *SIAM Review*, vol. 47, no. 1, pp. 135–161, 2005.
2. M. W. Berry and M. Browne: *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, Second Ed., SIAM, 2005.