Link analysis: PageRank

Web search results: desired

- List of webpages / websites ranked according to
  - Relevance to query
  - Importance / trustworthiness of websites - centrality
  - Location / time of query
  - Recency of page
  - … and many other factors

Node centrality in Web

- Web graph:
  - Nodes are webpages
  - Edges are hyperlinks (directed)

- We already discussed one algorithm for computing node centrality on the Web graph: HITS

- In this lecture, we see the most popular algorithm for node centrality on the Web

PAGERANK ALGORITHM

PageRank

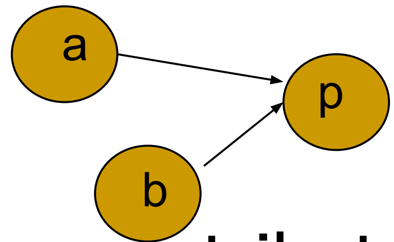
- By Larry Page and Sergey Brin

- PageRank of a page
  - Just one of many factors used by Google to rank pages
  - Independent of query

- Problem in measuring importance by indegree
  - Not all in-links are same
  - How important are those pages which link to page *p*?

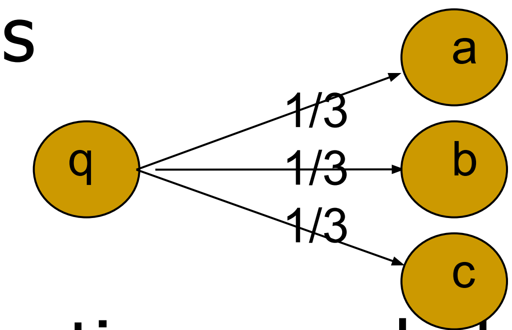
Idea of PageRank

- PR of page *p* is a function of the PR of pages which link to *p*



PR(p) is a function of PR(a) and PR(b)

- If page *q* links to 3 pages, *q* contributes *PR(q)/3* to the PR of each of those 3 pages



- Iterative algorithm, multiple iterations needed until convergence (similar to HITS)

PageRank computation

/* initialization */

for all nodes u in G: *d(u)* □ *1/N*, where *N* = #nodes

for all nodes u in G: *PR(u)* □ *d(u)*

/* iteration */

do until *PR* vector converges

    for all nodes *u* in G

        for all nodes *v* that links to *u*

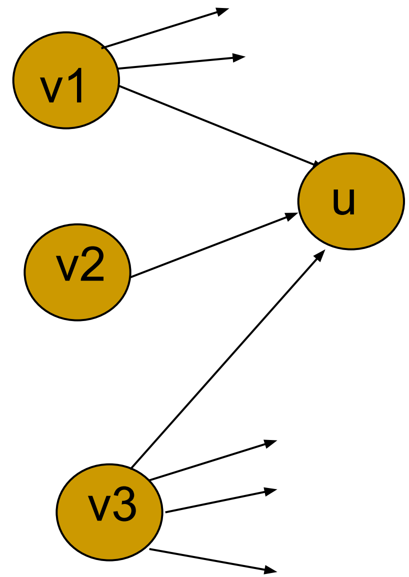
            $t = \Sigma\, PR(v) / \text{out-degree}(v)$

        PR(u) □ $\alpha * t + (1 - \alpha) * d(u)$

    normalize scores

    check for convergence

end

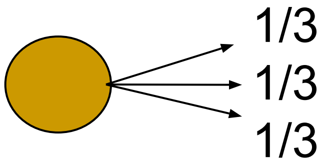


$t = PR(v1)/3 + PR(v2)/1 + PR(v3)/4$

α is a parameter; will be explained shor

Theoretical basis of PageRank

- Random surfer model
  - Start at a random node
  - Execute a random walk on Web graph
  - At each step, proceed from current node *u* to a randomly chosen node that *u* links to



- Random walk may reach a dead end
  - Teleport: jump to any random node with probability 1/N

Theoretical basis of PageRank

- Random surfer model
  - Start at a random node, and repeat this process:
  - At a node with no outgoing links (dead end), teleport
  - At a node that has outgoing links
    - Follow standard random walk with probability α where $0<\alpha<1$
    - Teleport with probability $(1-\alpha)$
  - Standard value of α: 0.85

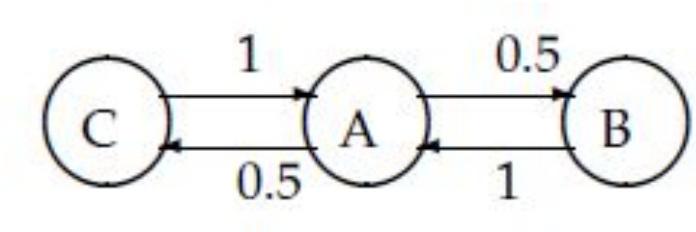
- Nodes visited more frequently in this random walk are web-pages with higher PR

Theoretical basis of PageRank

- The random walk defines a Markov chain
  - A discrete time stochastic process following Markov property (next state depends only on current state)
  - *N* states corresponding to the *N* nodes; the walk/Markov chain is at one of the states at any given time-step
  - *N* x *N* transition probability matrix *P* : P_{ij} is the probability that state at next time-step is *j*, given current state is *i*

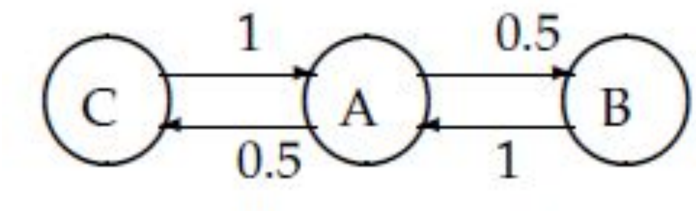
$$\forall i, j, P_{ij} \in [0,1] \qquad \forall i, \sum_{j=1}^{N} P_{ij} = 1.$$

Toy example of transition probability matrix



$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Toy example of transition probability matrix



$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

- *P* is a stochastic matrix
  - Every element is in [0, 1]
  - Sum of every row is 1
  - Largest eigenvalue is 1
  - Has a principal left eigenvector corresponding to its largest eigenvalue

Transition matrix for random surfer

- How to derive the transition matrix for the random surfer on the Web graph?

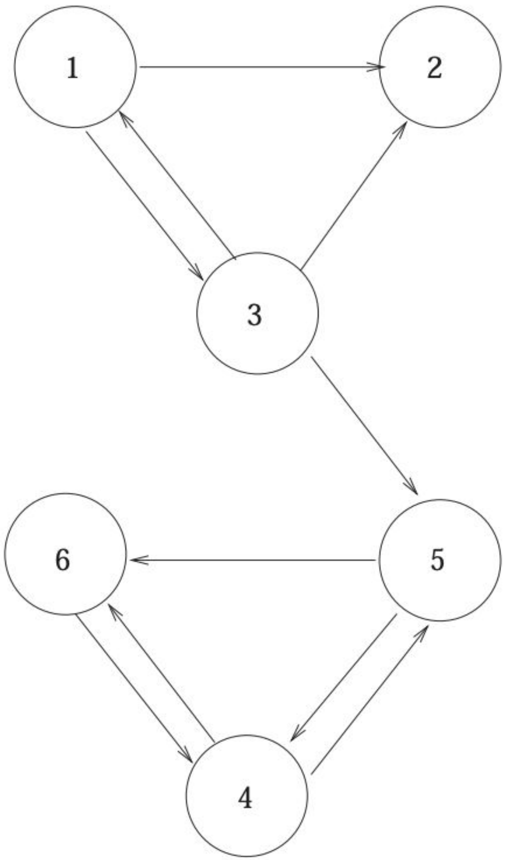
- Adjacency matrix of Web graph
  - $A_{ij} = 1$ if there is a hyperlink from page i to page j
  - $A_{ij} = 0$ otherwise

- Derive transition matrix P of Markov chain from A

Transition matrix for random surfer

- Derive transition matrix *P* of Markov chain from *A*
  - If a row of *A* has no 1's, replace each element by *1/N*
  - For all other rows: divide each 1 by the number of 1's in the row
  - Multiply the resulting matrix by α
  - Add (1-α)/*N* to every entry of the resulting matrix

Example: Mini web graph



$$\mathbf{P} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Example: Fixing sinks & teleporting

$$\bar{\mathbf{P}} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\bar{\bar{\mathbf{P}}} = \alpha\bar{\mathbf{P}} + (1-\alpha)\mathbf{e}\mathbf{e}^T/n = \begin{pmatrix} 1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\ 1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\ 1/60 & 1/60 & 1/60 & 11/12 & 1/60 & 1/60 \end{pmatrix}$$

Given P, how to compute PageRank?

- Vector x (dimension N): probability distribution of surfer's position at any time
  - At $t = 0$: one entry in x is 1, rest are 0
  - At $t = 1$: xP
  - At $t = 2$: $(xP)P = xP^2$
  - …
- Assume steady-state $x = \Pi$
  - Then $\Pi P = \Pi = 1.\Pi$
  - By definition, Π is the principal left eigenvector of P

Given *P*, how to compute PageRank?

- Hence PageRank scores obtained as the principal left eigenvector of *P*
- Corresponding to eigenvalue 1

PageRank computation

- Till now, we discussed two methods for computing PageRank
    1. Compute principal left eigenvector of a stochastic matrix derived from the adjacency matrix of the graph
    2. An iterative method (see slide 7)

- Several numerical methods available for computing eigenvectors of a matrix, e.g., power iteration
- Still, can be difficult for matrices of the size of the Web graph; iterative method can be more efficient

Why teleportation?

- Convergence of PageRank is guaranteed only if
  - The transition probability matrix P is irreducible, i.e., all transitions have a non-zero probability
  - In other words, if the graph (on which random surfing is taking place) is strongly connected

- To ensure convergence, conceptually do these:
  - From nodes with out-degree 0, add an outgoing edge to every node
  - Damp the walk by factor α, by adding a complete set of outgoing edges, with weight $(1-\alpha)/N$, to all nodes

Practical challenges

- All links *u*□*v* do not signify a vote for *v*
  - E.g., links to a copyright page from all pages in a website

- Attempts to spam PageRank: link spam farms or link farms
  - A target page (whose PR the spammer wants to boost)
  - A number of boosting pages, which link to the target page, link to each other and also to external pages
  - Hijacked links – links accumulated from pages outside the link farm

Example link farm

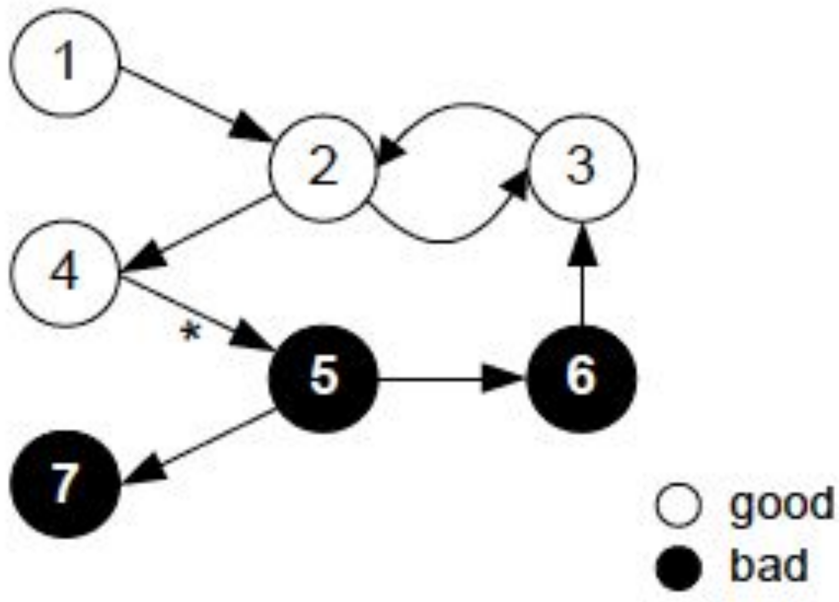


Figure 2: A web of good (white) and bad (black) nodes.

VARIATIONS OF PAGERANK

PageRank computation

/* initialization */

for all nodes u in G: *d(u)* □ *1/N*, where *N* = #nodes

for all nodes u in G: *PR(u)* □ *d(u)*

/* iteration */

do until *PR* vector converges

  for all nodes *u* in G

   for all nodes *v* that links to *u*

    $t = \Sigma\, PR(v)$ / out-degree(v)

   $PR(u)$ □ $\alpha * t + (1 - \alpha) * d(u)$

  normalize scores

  check for convergence

end

Biased PageRank

- Instead of using the uniform vector *d(u)* □ *1/N* for all nodes u, use a non-uniform preference vector:

  $d(u) \;=\; 1 \,/\, |S|$, for all $u \;\varepsilon\; S$

  $\qquad\;\, = \; 0$ otherwise

- The preference vector is said to be biased towards nodes in the subset S

Biased PageRank

- Instead of using the uniform vector *d(u)* □ *1/N* for all nodes u, use a non-uniform preference vector:

  d(u) = 1 / |S|, for all u ε S

  = 0 otherwise

- Implication for random surfer:
  - With probability α, follow standard random walk
  - With probability (1-α), teleport to a node in S, where the particular node in S is chosen randomly
- Ranks are biased towards nodes that are closer to nodes with a larger value in the preference vector

Topic-sensitive PageRank [Haveliwala, WWW 2002]

- Webpages are classified into various topics (16 Open Directory Project high-level categories)
- Goal is to compute PageRank, considering a particular category of interest

- For category c_j
  - T_j is the set of known websites for category c_j
  - Runs PageRank by biasing the preference vector towards the set of known websites in T_j
  - Expected: webpages relevant to the category of interest will be ranked higher

TrustRank [Gyongyi, VLDB 2004]

- Goal: rank trusted pages higher, and push untrusted pages down in the rankings
- Assumes:
  - Trusted (good) nodes are expected to only link to other good nodes, but this assumption is violated in the real Web
  - Bad nodes will link to other bad nodes and good nodes

- Assumes a way of knowing some trusted nodes
- Run PageRank by biasing the preference vector towards the set of trusted nodes

Conclusion

- Discussed two algorithms for identifying authoritative pages in the Web
  - HITS
  - PageRank
- Studied the theoretical basis of PageRank – Random Surfer model
- Brief discussion on some variants of PageRank