Page Rank and HITS

Page Rank

- The year 1998 was an important year for Web link analysis and Web search, as both the PageRank and the HITS algorithms were reported in that year.
- HITS was presented by Jon Kleinberg in January, 1998 at the *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. PageRank was presented by Sergey Brin and Larry Page at the *Seventh International World Wide Web Conference* (*WWW7*) in April, 1998.
- Based on the algorithm, they built the search engine Google.
- PageRank has emerged as the dominant link analysis model for Web search, partly due to its query-independent evaluation of Web pages and its ability to combat spamming, and partly due to Google's business success

Page Rank

- PageRank relies on the democratic nature of the Web by using its vast link structure as an indicator of an individual page's quality.

- In essence, PageRank interprets a hyperlink from page *x* to page *y* as a vote, by page *x*, for page *y*.

- However, It also analyzes the page that casts the vote. Votes casted by pages that are themselves “important” weigh more heavily and help to make other pages more “important.”

- This is exactly the idea of **rank prestige** in social networks.

Page Rank Algorithm

- PageRank is a static ranking of Web pages in the sense that a PageRank value is computed for each page off-line and it does not depend on search queries.
- Since PageRank is based on the measure of prestige in social networks, the PageRank value of each page can be regarded as its prestige.
- **In-links** of page *i*: These are the hyperlinks that point to page *i* from other pages. Usually, hyperlinks from the same site are not considered.
- A hyperlink from a page pointing to another page is an implicit conveyance of authority to the target page. Thus, the more in-links that a page *i* receives, the more prestige the page *i* has.

Page Rank Algorithm

- **Out-links** of page *i*: These are the hyperlinks that point out to other pages from page *i*. Usually, links to pages of the same site are not considered.
- Pages that point to page *i* also have their own prestige scores. A page with a higher prestige score pointing to *i* is more important than a page with a lower prestige score pointing to *i*. In other words, a page is important if it is pointed to by other important pages.
- According to rank prestige in social networks, the importance of page *i* (*i*'s PageRank score) is determined by summing up the PageRank scores of all pages that point to *i*. Since a page may point to many other pages, its prestige score should be shared among all the pages that it points to.

Page Rank Algorithm

- Let's treat the Web as a directed graph $G = (V, E)$, where V is the set of vertices or nodes, i.e., the set of all pages, and E is the set of directed edges in the graph, i.e., hyperlinks.
- Let the total number of pages on the Web be n (i.e., $n = |V|$).
- The PageRank score of the page i (denoted by $P(i)$) is defined by:

$$P(i) = \sum_{(j,i) \in E} \frac{P(j)}{O_j},$$

- where O_j is the number of out-links of page j.
- Let $\boldsymbol{P}$ be a n-dimensional column vector of PageRank values, i.e.,

$$\boldsymbol{P} = (P(1), P(2), \ldots, P(n))^T.$$

- Let $\boldsymbol{A}$ be the adjacency matrix of our graph with

$$A_{ij} = \begin{cases} \frac{1}{O_i} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Solve the PageRank equation

$$\boldsymbol{P} = \boldsymbol{A}^T \boldsymbol{P}$$

- This is the characteristic equation of the **eigensystem**, where the solution to ***P*** is an **eigenvector** with the corresponding **eigenvalue** of 1.
- It turns out that if ***some conditions*** are satisfied, 1 is the largest **eigenvalue** and the PageRank vector ***P*** is the **principal eigenvector**.
- A well known mathematical technique called **power iteration** can be used to find ***P***.
- **Problem**: the above Equation does not quite suffice because the Web graph does not meet the conditions.

Page Rank Algorithm

- In the Markov chain model, each Web page or node in the Web graph is regarded as a state.
- A hyperlink is a transition, which leads from one state to another state with a probability.
- Thus, this framework models Web surfing as a stochastic process.

Page Rank Algorithm

- It models a Web surfer randomly surfing the Web as a state transition in the Markov chain.

- Recall that we used O_i to denote the number of out-links of a node *i*.

- Each transition probability is $1/O_i$ if we assume the Web surfer will click the hyperlinks in the page *I* uniformly at random, the “back” button on the browser is not used and the surfer does not type in an URL.

Page Rank Algorithm

- Let ***A*** be the state transition probability matrix, a square matrix of the following format,

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdot & \cdot & \cdot & A_{1n} \\ A_{21} & A_{22} & \cdot & \cdot & \cdot & A_{2n} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ A_{n1} & A_{n2} & \cdot & \cdot & \cdot & A_{nn} \end{pmatrix}$$

- A_{ij} represents the transition probability that the surfer in state *i* (page *i*) will move to state *j* (page *j*).
- Given an **initial probability distribution** vector that a surfer is at each state (or page) $\boldsymbol{p}_0 = (p_0(1), p_0(2), \ldots, p_0(n))^T$ (a column vector) and an $n \times n$ **transition probability matrix *A***, we have

$$\sum_{i=1}^{n} p_0(i) = 1$$

$$\sum_{j=1}^{n} A_{ij} = 1 .$$

Back to the Markov chain

- In a Markov chain, a question of common interest is:
  - Given $\boldsymbol{p}_0$ at the beginning, what is the probability that *m* steps/transitions later the Markov chain will be at each state *j*?
- We determine the probability that the system (or the random surfer) is in state *j* after 1 step (1 transition) by using the following reasoning:

$$p_1(j) = \sum_{i=1}^{n} A_{ij}(1)\, p_0(i)$$

Stationary Probability Distribution

- The last Equation is not quite true for some Web pages because they have no out-links. If the matrix ***A*** satisfies that Equation, we say that ***A*** is the **stochastic matrix** of a Markov chain.

- By the Ergodic Theorem of Markov chains, a finite Markov chain defined by the **stochastic transition matrix** ***A*** has a unique **stationary probability distribution** if ***A*** is **irreducible** and **aperiodic**.

- Now let us come back to the real Web context and see whether the above conditions are satisfied, i.e., whether ***A*** is a stochastic matrix and whether it is irreducible and aperiodic. In fact, none of them is satisfied.

Page Rank Algorithm

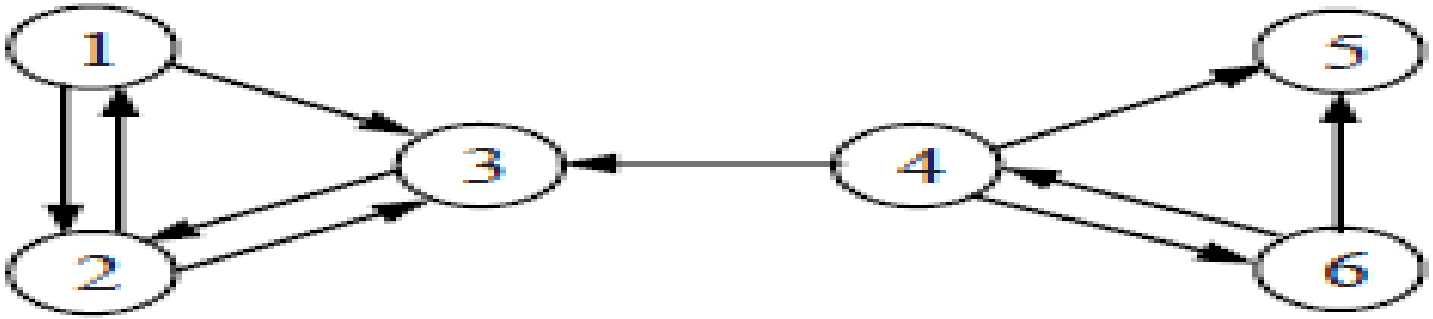
- First of all, ***A*** is not a **stochastic (transition) matrix**. A stochastic matrix is the transition matrix for a finite Markov chain whose entries in each row are non-negative real numbers and sum to 1.

- This requires that every Web page must have at least one out-link.

- This is not true on the Web because many pages have no out-links, which are reflected in transition matrix ***A*** by some rows of complete 0's.

- Such pages are called the **dangling pages** (nodes).

Page Rank Algorithm :: Example



- If we assume that the Web surfer will click the hyperlinks in a page uniformly at random, we have the following transition probability matrix:

$$A = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Page Rank Algorithm :: Example

- For example $A_{12} = A_{13} = 1/2$ because node 1 has two out-links.
- We can see that ***A*** is not a stochastic matrix because the fifth row is all 0's, i.e., page 5 is a dangling page.
- We can fix this problem in several ways in order to convert ***A*** to a stochastic transition matrix.
  - Remove those pages with no out-links from the system during the PageRank computation as these pages do not affect the ranking of any other page directly. Out-links from other pages pointing to these pages are also removed. After PageRanks are computed, these pages and hyperlinks pointing to them can be added in. The transition probabilities of those pages with removed links will be slightly affected but not significantly.

Page Rank Algorithm :: Example

- Add a complete set of outgoing links from each such page *i* to all the pages on the Web. Thus the transition probability of going from *i* to every page is 1/*n* assuming uniform probability distribution. That is, we replace each row containing all 0's with ***e***/*n*, where ***e*** is *n*-dimensional vector of all 1's.

- If we use the second method to make ***A*** a stochastic matrix by adding a link from page 5 to every page, we obtain

$$\overline{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Page Rank Algorithm :: Example

- ***A*** is not **irreducible**. Irreducible means that the Web graph *G* is strongly connected.
- **Definition** (**strongly connected**)**:** A directed graph $G = (V, E)$ is **strongly connected** if and only if, for each pair of nodes $u, v \in V$, there is a path from u to v.
- A general Web graph represented by ***A*** is not irreducible because for some pair of nodes u and v, there is no path from u to v.
- For example, in the example, there is no directed path from node 3 to node 4. That is, in $\overline{A}$ there is still no directed path from node 3 to node 4.

A is a not aperiodic

- A state *i* in a Markov chain being periodic means that there exists a directed cycle that the chain has to traverse.

Definition: A state *i* is **periodic** with period $k > 1$ if *k* is the smallest number such that all paths leading from state *i* back to state *i* have a length that is a multiple of *k*.

  - If a state is not periodic (i.e., $k = 1$), it is **aperiodic**.
  - A Markov chain is **aperiodic** if all states are aperiodic.

An example: periodic

- Fig. 5 shows a periodic Markov chain with $k = 3$. Eg, if we begin from state 1, to come back to state 1 the only path is 1-2-3-1 for some number of times, say h. Thus any return to state 1 will take $3h$ transitions.

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

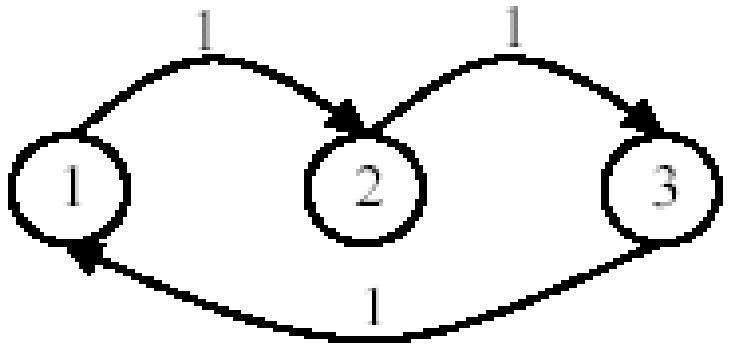


Fig. 5. A Periodic Markov chain with $k = 3$.

Deal with irreducible and aperiodic

- It is easy to deal with the above two problems with a single strategy.
- Add a link from each page to every page and give each link a small transition probability controlled by a parameter *d*.
- Obviously, the augmented transition matrix becomes **irreducible** and **aperiodic**

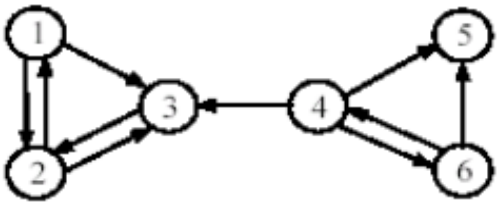
Improved PageRank

- After this augmentation, at a page, the random surfer has two options
  - With probability *d*, he randomly chooses an out-link to follow.
  - With probability 1-*d*, he jumps to a random page
- Equation gives the improved model,

$$\boldsymbol{P} = ((1-d)\frac{\boldsymbol{E}}{n} + d\boldsymbol{A}^T)\boldsymbol{P}$$

where $\boldsymbol{E}$ is $\boldsymbol{e}\boldsymbol{e}^T$ ($\boldsymbol{e}$ is a column vector of all 1's) and thus $\boldsymbol{E}$ is a $n \times n$ square matrix of all 1's.

Follow our example



The matrix made *stochastic*, which is still:
- *periodic* (see state 3)
- *reducible* (no path from 3 to 4)

$$A = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

Transposed matrix

$d = 0.9$

$$(1-d)\frac{\boldsymbol{E}}{n} + dA^T = \begin{pmatrix} 1/60 & 7/15 & 1/60 & 1/60 & 1/6 & 1/60 \\ 7/15 & 1/60 & 11/12 & 1/60 & 1/6 & 1/60 \\ 7/15 & 7/15 & 1/60 & 19/60 & 1/6 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 1/60 \end{pmatrix}$$

The final PageRank algorithm

- **Given:**

$$\boldsymbol{P} = (1-d)\boldsymbol{e} + d\boldsymbol{A}^T\boldsymbol{P}$$

$$A_{ji} = \begin{cases} \frac{1}{O_j} & if\ (j,i) \in E \\ 0 & otherwise \end{cases}$$

PageRank for each page *i* is:

$$P(i) = (1-d) + d\sum_{j=1}^{n} A_{ji}P(j)$$

that is equivalent to the formula given in the PageRank paper [BP98]

- **The parameter d is called the damping factor which can be set to between 0 and 1. $d = 0.85$ was used in the PageRank paper**

$$P(i) = (1-d) + d\sum_{(j,i)\in E} \frac{P(j)}{O_j}$$

[BP98] Sergey Brin and Lawrence Page. **The Anatomy of a Large-Scale Hypertextual Web Search Engine.** WWW Int.l Conf., 1998.

Compute PageRank

- Use the power iteration method

PageRank-Iterate(G)

$P_0 \leftarrow e/n$

$k \leftarrow 1$

repeat

$P_k \leftarrow (1-d)e + dA^T P_{k-1}$;

$k \leftarrow k+1$;

until $||P_k - P_{k-1}||_1 < \varepsilon$

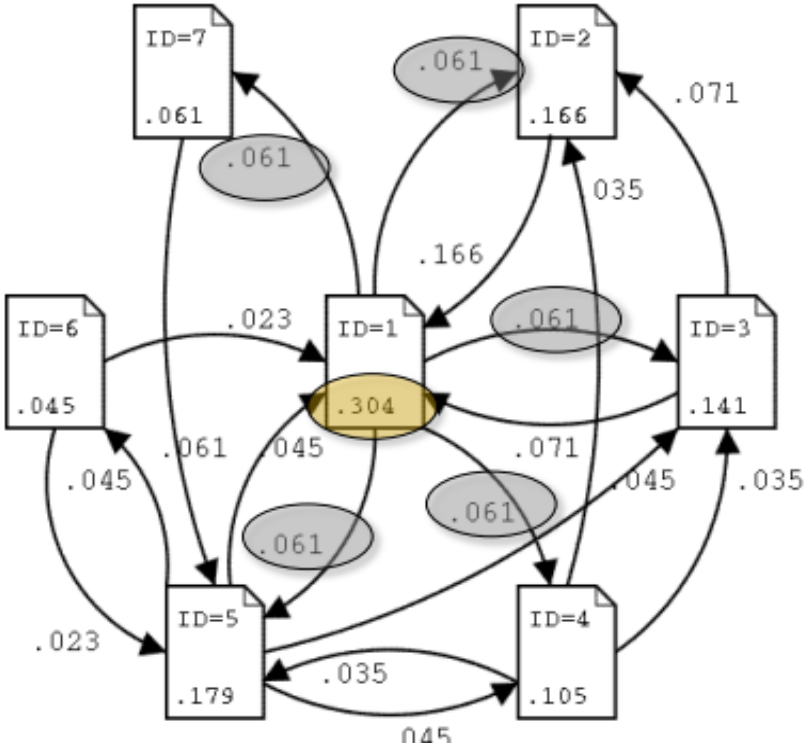
return P_k

Fig. 7.7. The power iteration method for PageRank

Compute PageRank :: Example

- **Without scaling the equation (by multiplying by n), we have $e^T P = 1$ (i.e., the sum of all PageRanks is one), and thus:**

$$P(i) = \frac{1-d}{n} + d \sum_{(j,i)\in E} \frac{P(j)}{O_j}$$

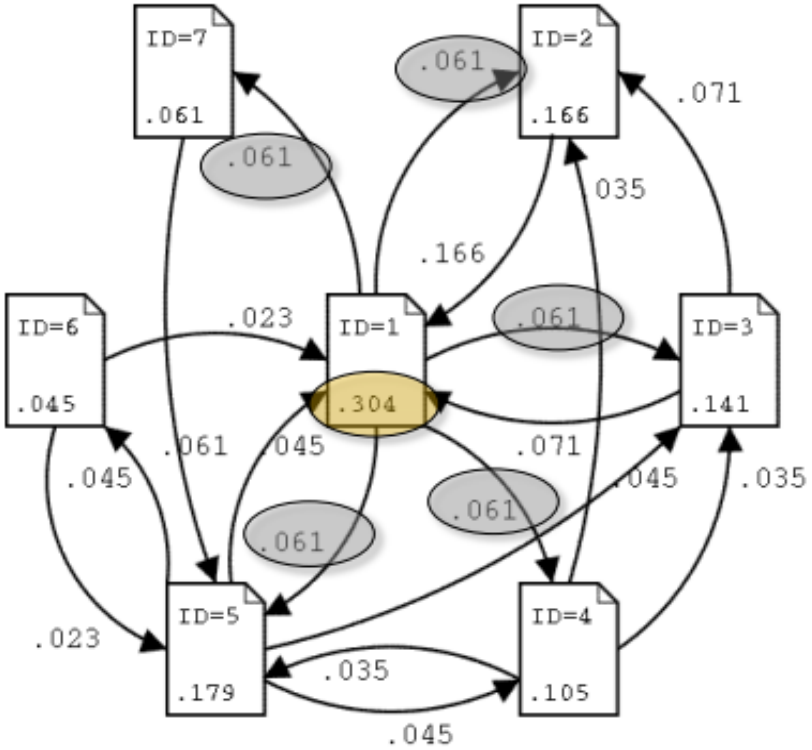


- **Important pages**
  - **are cited/pointed by other important ones**
- **In the example, the most important is ID=1**
  - **P(ID=1) = 0.304**
- **P(ID=1) distributes is "rank" among all its 5 outgoing links**
  - **ID= 2, 3, 4, 5, 7**
  - **0.304 = 0.061 * 5**

Compute PageRank :: Example

- **Without scaling the equation (by multiplying by n), we have $e^T P = 1$ (i.e., the sum of all PageRanks is one), and thus:**

$$P(i) = \frac{1-d}{n} + d \sum_{(j,i)\in E} \frac{P(j)}{O_j}$$



- **The stationary probability P(ID=1) is obtained by:**

  (1-d)/n +
  d (0.023+0.166+0.071+0.045)=

  (0.15)/7 +
  0.85(0.023+0.166+0.071+0.045)=

  0.304

Advantages of PageRank

- **Fighting spam**. A page is important if the pages pointing to it are important.
  - Since it is not easy for Web page owner to add in-links into his/her page from other important pages, it is thus not easy to influence PageRank.
- **PageRank is a global measure and is query independent**.
  - PageRank values of all the pages are computed and saved off-line rather than at the query time.
- Criticism: Query-independence. It could not distinguish between pages that are authoritative in general and pages that are authoritative on the query topic.

HITS

- HITS stands for **Hypertext Induced Topic Search**.

- Unlike PageRank which is a static ranking algorithm, HITS is search query dependent.

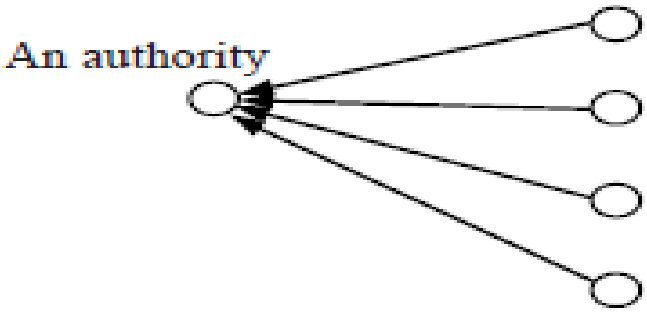
- When the user issues a search query,
  - HITS first expands the list of relevant pages returned by a search engine and
  - then produces two rankings of the expanded set of pages, **authority ranking** and **hub ranking**.

HITS

- **Authority**: Roughly, a authority is a page with many in-links.
  - The idea is that the page may have good or authoritative content on some topic and
  - thus many people trust it and link to it.

- **Hub**: A hub is a page with many out-links.
  - The page serves as an organizer of the information on a particular topic and
  - points to many good authority pages on the topic.

HITS



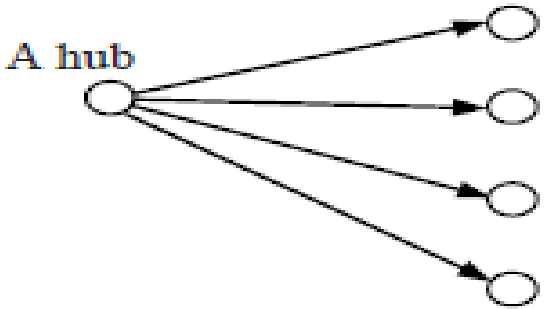


Fig. 7.8. An authority page and a hub page

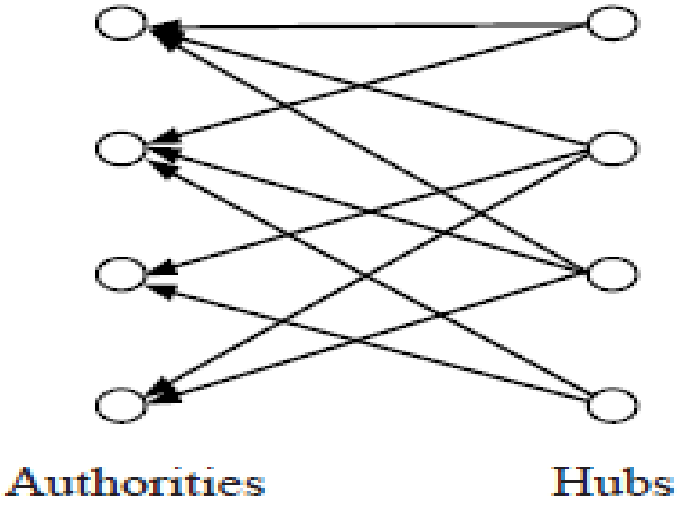


Fig. 7.9. A densely linked set of authorities and hubs

HITS

- When a user comes to this hub page, he/she will find many useful links which take him/her to good content pages on the topic.
- The key idea of HITS is that a good hub points to many good authorities and a good authority is pointed to by many good hubs.

- Thus, authorities and hubs have a **mutual reinforcement** relationship.
- Fig. 7.9 shows a set of densely linked authorities and hubs (a **bipartite sub-graph**).

- Bipartite graph is a graph, who vertices can be divided into two disjoint and independent sets U and V s. t. every edge connects a vertex in U to one in V

HITS Algorithm

- Given a broad search query, *q*, HITS collects a set of pages as follows:
  - It sends the query *q* to a search engine.
  - It then collects *t* (*t* = 200 is used in the HITS paper) highest ranked pages. This set is called the **root** set *W*.
  - It then grows *W* by including any page pointed to by a page in *W* and any page that points to a page in *W*. This gives a larger set *S*, **base set**.

HITS Algorithm

- HITS works on the pages in *S*, and assigns every page in *S* an **authority score** and a **hub score**.

- Let the number of pages in *S* be *n*.

- We again use *G* = (*V*, *E*) to denote the hyperlink graph of *S*.
- We use ***L*** to denote the adjacency matrix of the graph.

$$L_{ij} = \begin{cases} 1 & if\ (i, j) \in E \\ 0 & otherwise \end{cases}$$

HITS Algorithm

- Let the authority score of the page *i* be *a*(*i*), and the hub score of page *i* be *h*(*i*).
- The mutual reinforcing relationship of the two scores is represented as follows:

$$a(i) = \sum_{(j,i)\in E} h(j) \qquad h(i) = \sum_{(i,j)\in E} a(j)$$

- We use ***a*** to denote the column vector with all the authority scores,

  a = (a(1), a(2), …, a(n))^T, and

- use ***h*** to denote the column vector with all the hub scores,

  h = (h(1), h(2), …, h(n))^T,

- Then,

$$\boldsymbol{a} = \boldsymbol{L}^T\boldsymbol{h} \qquad \boldsymbol{h} = \boldsymbol{L}\boldsymbol{a}$$

HITS Algorithm

- The computation of authority scores and hub scores is the same as the computation of the PageRank scores, using power iteration.
- If we use $\boldsymbol{a}_k$ and $\boldsymbol{h}_k$ to denote authority and hub vectors at the kth iteration, the iterations for generating the final solutions are

$$\boldsymbol{a}_k = \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{a}_{k-1}$$

$$\boldsymbol{h}_k = \boldsymbol{L}\boldsymbol{L}^T\boldsymbol{h}_{k-1}$$

starting with

$$\boldsymbol{a}_0 = \boldsymbol{h}_0 = (1, 1, \ldots, 1).$$

HITS Algorithm

HITS-Iterate(G)

$\boldsymbol{a}_0 \leftarrow \boldsymbol{h}_0 \leftarrow (1, 1, \ldots, 1)$;

$k \leftarrow 1$

Repeat

$\boldsymbol{a}_k \leftarrow \boldsymbol{L}^T \boldsymbol{L} \boldsymbol{a}_{k-1}$;

$\boldsymbol{h}_k \leftarrow \boldsymbol{L}\boldsymbol{L}^T \boldsymbol{h}_{k-1}$;

$\boldsymbol{a}_k \leftarrow \boldsymbol{a}_k / ||\boldsymbol{a}_k||_1$; // normalization

$\boldsymbol{h}_k \leftarrow \boldsymbol{h}_k / ||\boldsymbol{h}_k||_1$; // normalization

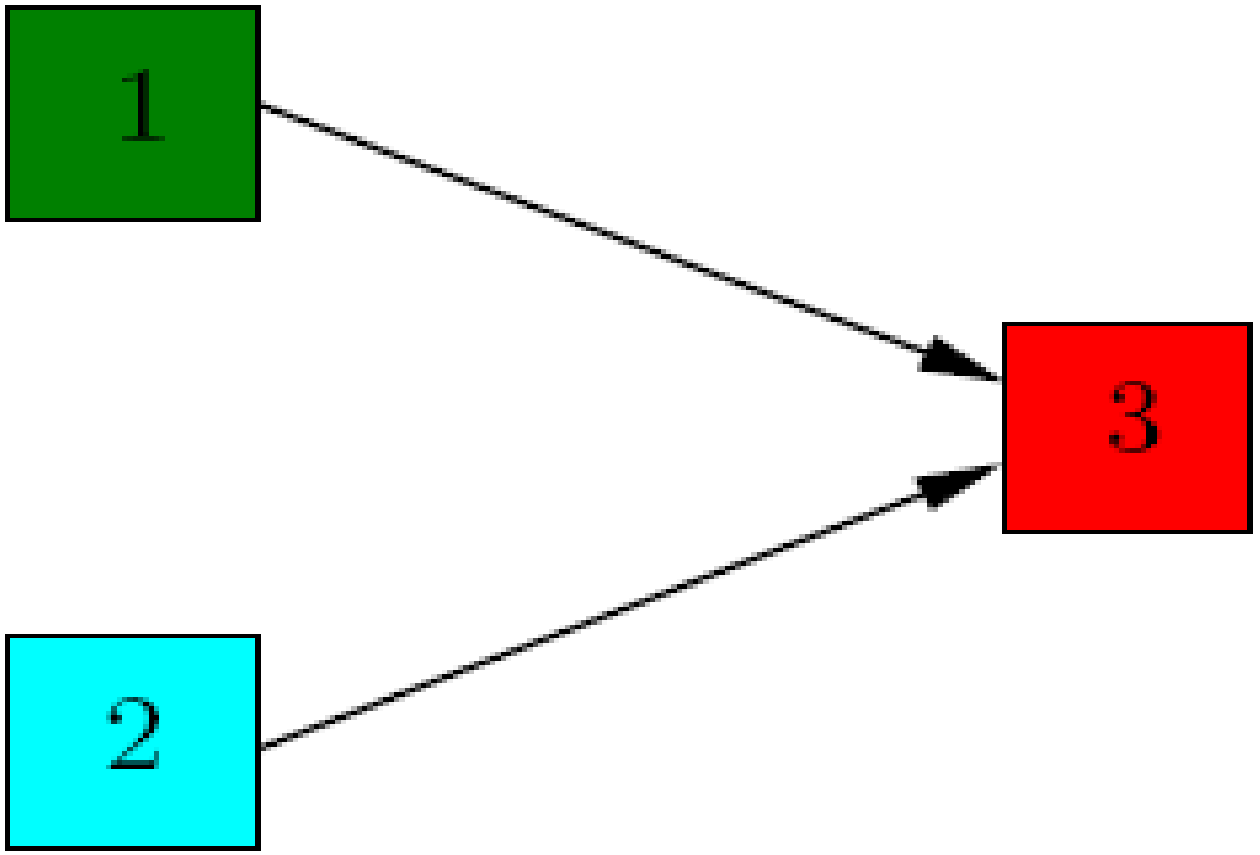
$k \leftarrow k + 1$;

until $||\boldsymbol{a}_k - \boldsymbol{a}_{k-1}||_1 < \varepsilon_a$ and $||\boldsymbol{h}_k - \boldsymbol{h}_{k-1}||_1 < \varepsilon_h$;

return $\boldsymbol{a}_k$ and $\boldsymbol{h}_k$

Fig. 7.10. The HITS algorithm based on power iteration

HITS Algorithm : Practice Sample Example



HITS Algorithm : Practice Sample Example

The adjacency matrix of the graph is $A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$, with transpose $A^t = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$. Assume the initial hub weight vector is: $u = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

We compute the authority weight vector by:

$$v = A^t \cdot u = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

Then, the updated hub weight is:

$$u = A \cdot v = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}$$

HITS Algorithm : Practice Sample Example