Link Analysis

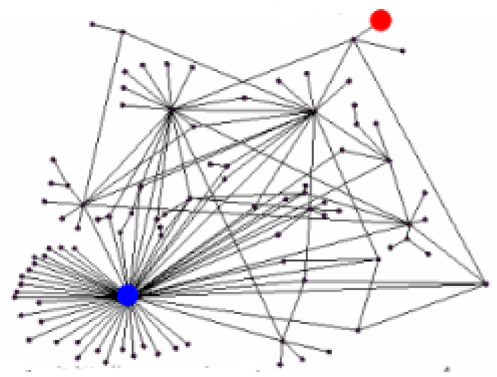
Link analysis is a data analysis technique used in network theory that is used to evaluate the relationships or connections between network nodes. These relationships can be between various types of objects (nodes), including people, organizations and even transactions.

Link analysis is essentially a kind of knowledge discovery that can be used to visualize data to allow for better analysis, especially in the context of links, whether Web links or relationship links between people or between different entities. Link analysis is often used in search engine optimization as well as in intelligence, in security analysis and in market and medical research.

For what concerns Web Search, different approaches have been studied previously to Web Links Analysis:

1. Human curated **Web directories** [Yahoo, DMOZ, LookSmart] - rapidly abandoned due to huge number of pages and links, impossible to mantain manually
2. **Web Search**: **information retrieval** attempts to find relevant docs in a small and trusted set (newspaper articles, patents, etc.) but there is a need for a good way to rank webpages.

Web Search has its own challenges: the definition of a **trustworthy** page, and the **best answer to a query**. Moreover, web pages are not equally important, in terms of references: they can be represented by the web graph link structure, which can be used to rank the pages.



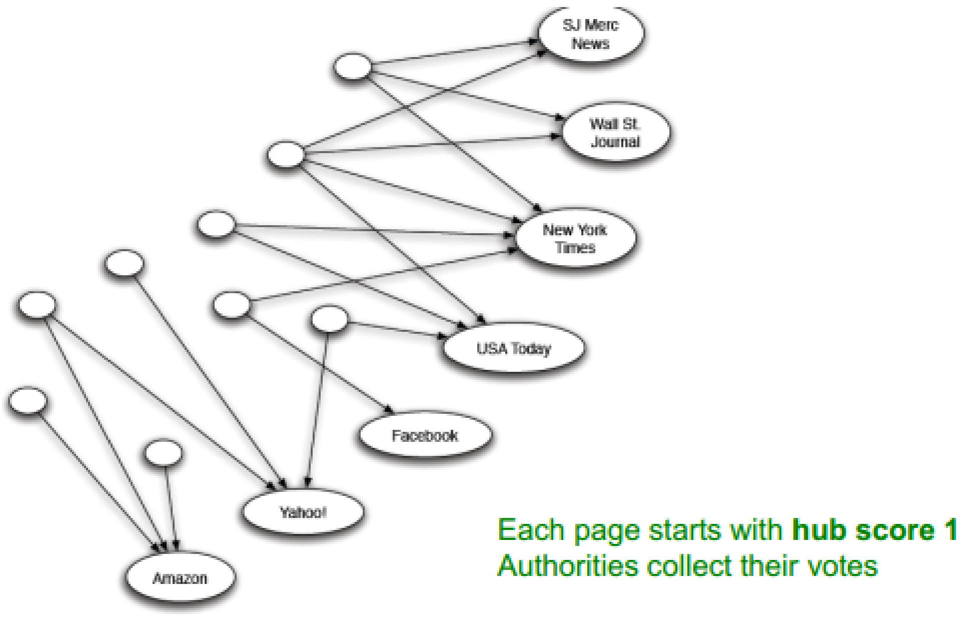
The HITS model

The HITS Model is based on the concept of **links as votes**: a page is more important is it has more links. However, it could be either in-links or out-links. In the HITS model, for a link from *i* to *j*, the value of the link depends on the links **out of *i*** .

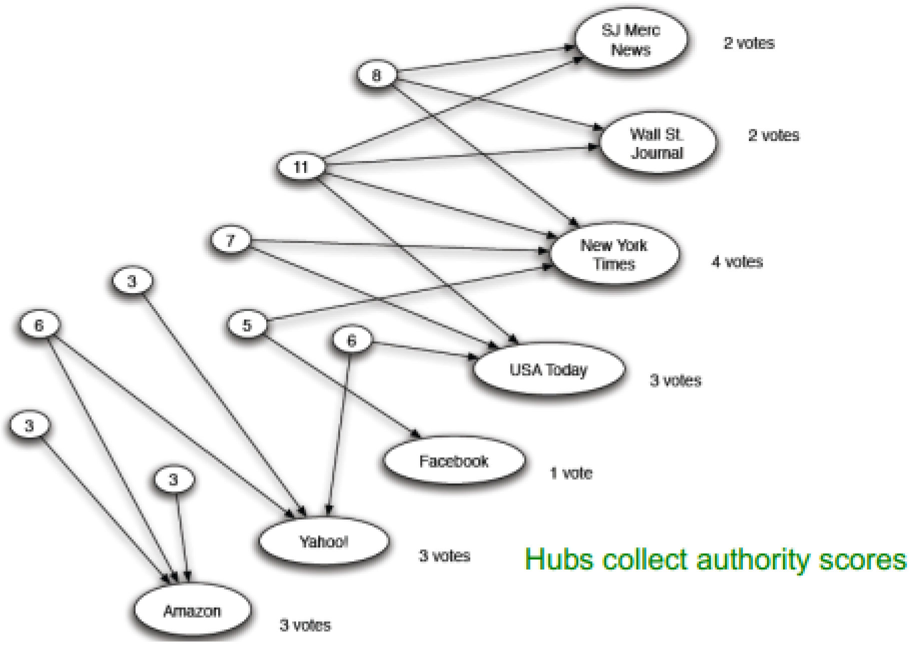
The model assigns two scores to a page:

- quality as an expert (**hub**) - total sum of votes of pages pointed to - a hub is a page that links to authorities
- quality as a content provider (**authority**) - total sum of votes of experts - an authority is a page that contains useful information

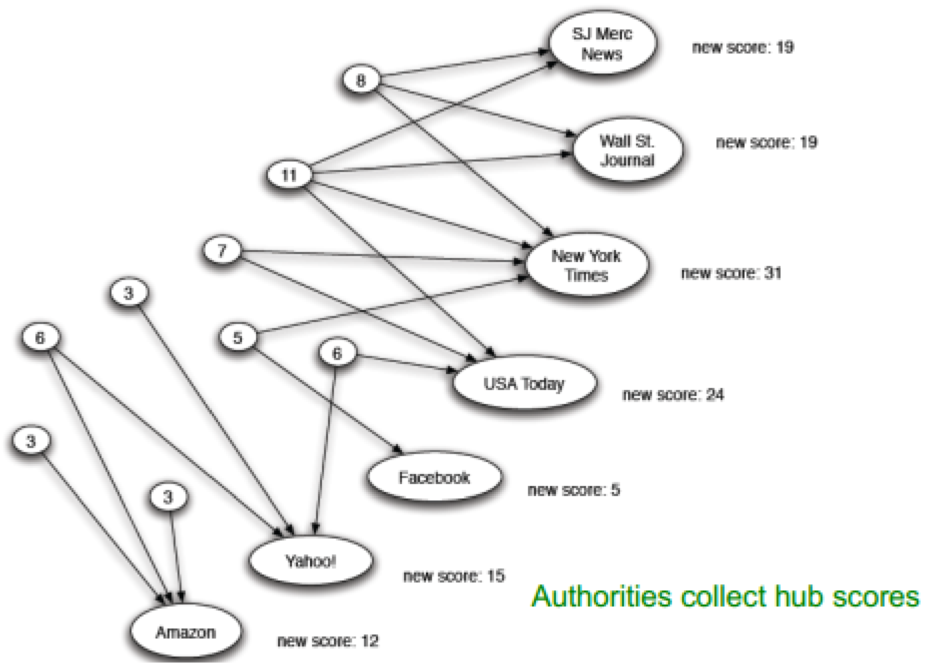
Let's analyse the following example graph:



Each page (hubs, circles without name) start with a **hub score** of 1 (in reality, it's 1/sqrt(n)). Authorities collect their votes, which means that, for example, Amazon will have a **authority score** of 3 because it is references by 3 pages, while the NYT page will have a authority score of 4. Then, the hubs collect the authority score, updating their hub scores. The following picture represent how hubs scores are updated.



Once again, authorities collect hub scores, updating their authority score with the sum of the hub scores referencing the authority page.



Therefore, some conclusion can be obtained:

- a **good hub** links to many good authorities
- a **good authority** is linked by many good hubs

The model is based on a self-reinforcing recursive definition! Good authorities will make better hubs, which will improve the very authorities that made them better.

The scores can be represented by two vector, *h* and *a*, where the *i-th* element is the hub/authority score of the i-th node. The algorithm then becomes:

- Initialize: $a_j^{(0)} = 1/\sqrt{\mathrm{n}}, \ \ \mathrm{h}_\mathrm{j}^{(0)} = 1/\sqrt{\mathrm{n}}$
- Then keep iterating until **convergence**:
  - $\forall \boldsymbol{i}$: Authority: $a_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}$
  - $\forall \boldsymbol{i}$: Hub: $h_i^{(t+1)} = \sum_{i \to j} a_j^{(t)}$
  - $\forall \boldsymbol{i}$: Normalize:

    $\sum_i \left(a_i^{(t+1)}\right)^2 = 1, \sum_j \left(h_j^{(t+1)}\right)^2 = 1$

where convergence can be the usual non-improvement criterion with a parameter *epsilon*:

Convergence criteria:

$$\sum_i \left(h_i^{(t)} - h_i^{(t+1)}\right)^2 < \varepsilon$$

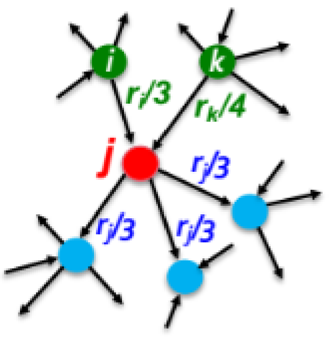
$$\sum_i \left(a_i^{(t)} - a_i^{(t+1)}\right)^2 < \varepsilon$$

PageRank

The PageRank algorithm is based on the concept of **links as votes**: a page is more important if it has more links. However, it could be either in-links or out-links. In the PageRank model, for a link from *i* to *j*, the value of the link depends on the links **into *i*** .

Let's consider in-links. If an "important" page references (links to) another page, that link should weight more, proportionally to the importance of the source page. So some definitions can be derived from this:

> If a page *i* with importance r_i has d_i out-links, each link gets r_i/d_i votes.
>
> Page *j*'s importance r_j is the sum of the votes on its in-links.



$$r_j = r_i/3 + r_k/4$$

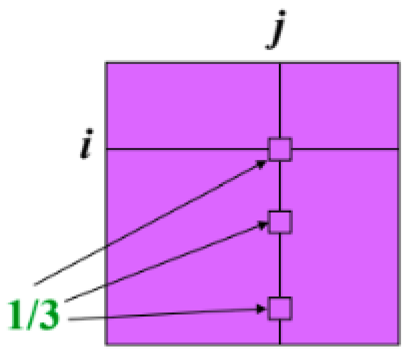
Therefore, a page is important **if it is pointed to by other important pages**. From the in-links, it is possible to define a **rank r_j** for node *j* as such:

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

d_i … out-degree of node i

Matrix intepretation

Let page *j* have d_j out-links. We can build a **column stochastic matrix M[i*j]** (AKA a matrix where the columns sum to 1) based on the rule that if there is a link **j --> i**, then $\mathbf{M_{ij}=1/d_j}$.



Random Walk interpretation

Let's imagine a random web surfer, which at a time *t* is on some page *i*. At time *t+1*, the surfer follows an out-link from *i* uniformly at random, ending on a page *j* pointed by *i*, and so on indefinitely.

We can define a vector *p(t)* as a probability distribution over pages, which means that it is made of probabilities that the surfer is at some page (its index) at time *t*.

At time *t+1*, the surfer moves to a different page. The vector will then become:

> *p(t+1) = M p(t) = p(t)*

This means that *p(t)* is a **stationary distribution** of a random walk and, according to the previous definition of *rank vector r*, the rank vector *r* itself is a stationary distribution for the random walk.

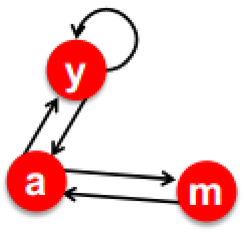
Solving the PageRank problem

As previously said, the web surfer indefinitely keeps on surfing web pages. Let's assign each node (a web page) an initial page rank, equal to *1/N*, where N is the number of nodes. The rank itself is updated at time *t+1* by computing the *rank* just like before:

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}}$$

This process repeats until convergence, which means for example that there is no visible improvement in the ranking at time *t+1*.

An easy example is the following:



1. every page *j* has rank r_j*=1/3* . Let's consider page *a*
2. $r'_a = sum_{i\text{->}a}(r_i/d_i) = (1/3)/2 + 1/3 = 1/2$
   - NB: i = {j, m} here
3. *r = r'*
4. repeat until convergence - if |*r-r'*|>*epsilon*, goto 2

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

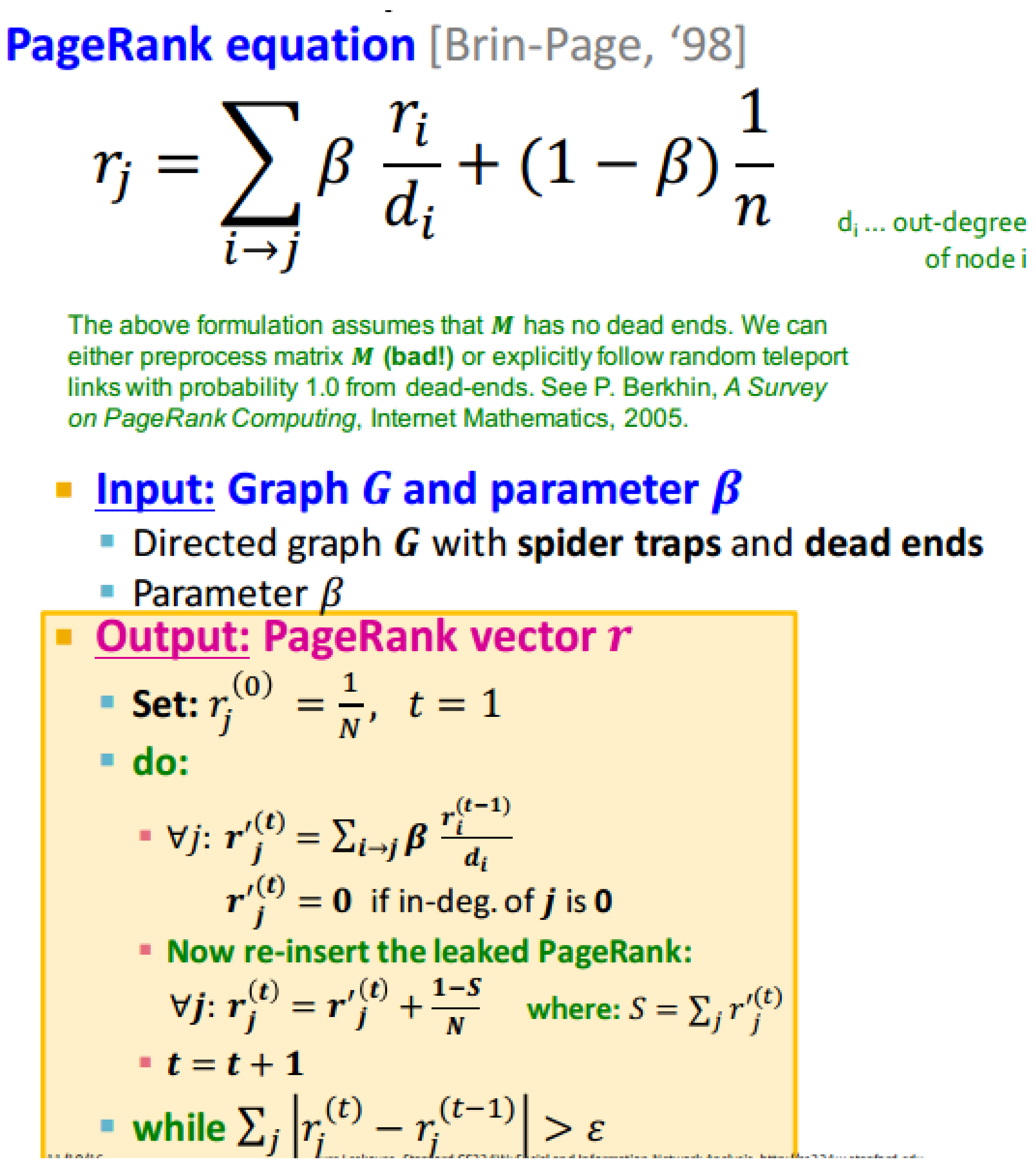
Problems

Sometimes convergence for the pagerank problem can be tricky. Two problems exist:

- **dead ends** - some pages have no out-link - values converge to 0

- **spider traps** - all out-links refer to the page itself - the page with the spider trap absorb all the importance

These problems can be resolved with **random teleports**: at each time step, the random surfer can either follow a link at random with probability *0.8 < b < 0.9*, or jump to a random page with probability *1-b*. The probability *b* is set to 1 for dead-ends, which means that the surfer always teleports out of the dead-end. The final equation and algorithm become:



PageRank equation [Brin-Page, '98]

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

d_i … out-degree of node i

The above formulation assumes that M has no dead ends. We can either preprocess matrix M **(bad!)** or explicitly follow random teleport links with probability 1.0 from dead-ends. See P. Berkhin, *A Survey on PageRank Computing*, Internet Mathematics, 2005.

- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}$, $t = 1$
  - **do:**
    - $\forall j$: $r'^{(t)}_j = \sum_{i \to j} \beta \frac{r_i^{(t-1)}}{d_i}$

      $r'^{(t)}_j = 0$ if in-deg. of j is 0
    - **Now re-insert the leaked PageRank:**

      $\forall j$: $r_j^{(t)} = r'^{(t)}_j + \frac{1-S}{N}$ where: $S = \sum_j r'^{(t)}_j$
    - $t = t + 1$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$