Web Algorithms – Web Search

Part 2: PageRank

Eng. Fabio Persia, PhD

PageRank

- Introduced in the SE at Stanford University by Larry Page and Sergey Brin in late 1996 (known from the 60s!)
- In PageRank the quality of a page is determined by the number of incoming links in it and, recursively, by the quality of the pages that link
- The algorithm is based on the model of "random Web surfer or without purpose": consider a Web surfer who eternally click on the hyperlink, choosing uniformly on each page randomly a link to move to the next page
- We assume for the moment that the (directed) Web graph is strongly connected, or that for each node ***u*** there is a direct path to the node ***v***

PageRank

- Suppose now that the surfer starts from a random node according to the probability distribution $\boldsymbol{p_0}$, i.e. $\boldsymbol{p_0[u]}$ is the probability to start from the node ***u***. $(\sum_u \boldsymbol{p_0[u]=1})$
- The navigator can follow any outgoing links from a page with equal probability.
- For convenience we denote as ***E*** the Web adjacency matrix, i.e. such that ***E[u,v]=1*** if there is an edge from node ***u*** to node ***v***, ***E[u,v]=0*** otherwise

PageRank

- The calculation of PageRank corresponds to the calculation of the principal eigenvector of a matrix derived from the adjacency matrix

- After clicking once, what is the probability $p_1[v]$ that the navigator is on the page ***v***?

In order to reach ***v***, in the previous step it must have been in some node ***u*** that would contain a link to ***v***

Given E, the ***<u>out-degree</u>*** of ***u*** N_u node (the number of outgoing links from ***u***) is simply given by the sum of the entries of ***E*** in the row corresponding to ***u***, i.e.

$$N_u = \sum_v E[u,v]$$

PageRank

- Assuming that you cannot have parallel arches (multiple links from ***u*** to ***v***), the probability of the last event given the previous (i.e. be in ***u***), is exactly ***1/N$_u$***

- Combining, $$p_1[v] = \sum_{u|(u,v)\in E} \frac{p_0[u]}{N_u}$$

- And now we derive from ***E*** a matrix ***L***, normalizing all the amounts of each line to one, that is,

$$L[u,v] = \frac{E[u,v]}{\sum_w E[u,w]} = \frac{E[u,v]}{N_u}$$

PageRank

- ***$p_1[v]$*** can be described as

$$p_1[v] = \sum_u L[u,v]p_0[u]$$

or as

$$p_1 = L^T p_0$$

- Iterating, after the i-th step, we get

$$p_{i+1} = L^T p_i$$

- We will assume initially that the nodes that have no outgoing links have been removed in advance

PageRank

- We assume that ***E*** (and thus ***L***) are
  - ***Irreducible***: there is a direct path from each node to every other node
  - ***Aperiodic***: for each ***(u, v)*** there are paths that connect them with each possible number of links, every possible length, with the exception of a finite set of lengths which may be missing

- Then the sequence ***(p_i), i = 0,1,2 ...,*** converge to a principal eigenvector $\boldsymbol{L^T}$, that is, to a solution of the matrix equation

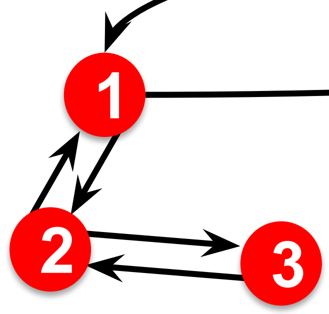
$$p = L^T p$$

- It is also called stationary distribution of L and is independent of p_0!

- Such prestige of node ***u***, denoted by ***p[u]***, is said to be its ***PageRank***

PageRank: How to solve?

- **Power Iteration:**
  - Set $p_j = 1/N$
  - $p'_j = \sum_{i \to j} \frac{p_i}{N_i}$
  - $p = p'$
  - Iterate



$$L^T = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & ½ & ½ & 0 \\ 2 & ½ & 0 & 1 \\ 3 & 0 & ½ & 0 \end{array}$$

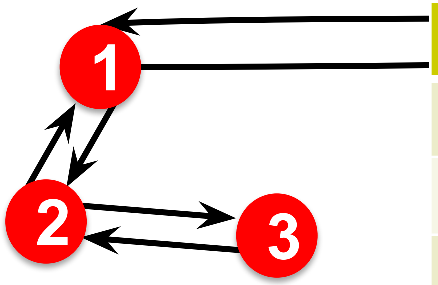
- **Example:**

$$\begin{aligned} p_1 &= p_1/2 + p_2/2 \\ p_2 &= p_1/2 + p_3 \\ p_3 &= p_2/2 \end{aligned}$$

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

PageRank: How to solve?

- **Power Iteration:**
  - Set $p_j = 1/N$
  - **1:** $p'_j = \sum_{i \to j} \frac{p_i}{N_i}$
  - $p = p'$
  - Iterate



	1	2	3
1	½	½	0
2	½	0	1
3	0	½	0

$$p_1 = p_1/2 + p_2/2$$
$$p_2 = p_1/2 + p_3$$
$$p_3 = p_2/2$$

- **Example:**

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{array}{cccccc} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{array}$$

Iteration 0, 1, 2, …

Why Power Iteration works?

- **Power iteration:**
  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - $p_1 = L^T \cdot p_o$
  - $p_2 = L^T \cdot p_1 = L^T(L^T p_1) = L^T{}_2 \cdot p_o$
  - $p_3 = L^T \cdot p_2 = L^T(L^T{}_2 p_2) = L^T{}_3 \cdot p_o$

- **Claim:**
  Sequence $L^T \cdot p_o, L^T{}_2 \cdot p_o, \ldots, L^T{}_k \cdot p_o, \ldots$
  approaches the dominant eigenvector of L^T

Why Power Iteration works?

- **Claim:** Sequence $\boldsymbol{L}^T \cdot \boldsymbol{p_0}, \boldsymbol{L_2}^T \cdot \boldsymbol{p_0}, \ldots \boldsymbol{L_k}^T \cdot \boldsymbol{p_0}, \ldots$ approaches the dominant eigenvector of $\boldsymbol{L}^T$
- **Proof:**
  - Assume $\boldsymbol{L}^T$ has ***n*** linearly independent eigenvectors, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_1 > \lambda_2 > \cdots > \lambda_n$
  - Vectors $x_1, x_2, \ldots, x_n$ form a basis and thus we can write: $p_0 + c_2\, x_2 + \cdots + c_n\, x_n$
  - $\boldsymbol{L}^T \boldsymbol{p_0} = \boldsymbol{L}^T(\boldsymbol{c_1\, x_1 + c_2\, x_2 + \cdots + c_n\, x_n})$

    $$= c_1(\boldsymbol{L}^T x_1) + c_2(\boldsymbol{L}^T x_2) + \cdots + c_n(\boldsymbol{L}^T x_n)$$

    $$= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \cdots + c_n(\lambda_n x_n)$$
  - **Repeated multiplication on both sides produces**

    $$\boldsymbol{L}_k{}^T p_0 = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \cdots + c_n(\lambda_n^k x_n)$$

Why Power Iteration works?

- **Claim:** Sequence $\boldsymbol{L}^T \cdot \boldsymbol{p}_0, \boldsymbol{L}_2{}^T \cdot \boldsymbol{p}_0, \dots \boldsymbol{L}_k{}^T \cdot \boldsymbol{p}_0, \dots$ approaches the dominant eigenvector of $\boldsymbol{L}^T$

- **Proof (continued):**
  - Repeated multiplication on both sides produces
  $$\boldsymbol{L}_k{}^T \boldsymbol{r}^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \dots + c_n(\lambda_n^k x_n)$$
  - $\boldsymbol{L}_k{}^T\, r^{(0)} = \lambda_1^k \left[c_1 x_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \dots + c_n \left(\frac{\lambda_2}{\lambda_1}\right)^k x_n \right]$
  - Since $\lambda_1 > \lambda_2$ then fractions $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1} \dots < 1$

    and so $\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$ as $k \to \infty$ (for all $i = 2 \dots n$).
  - **Thus:** $\boldsymbol{L}_k{}^T r^{(0)} \approx \boldsymbol{c_1(\lambda_1^k x_1)}$
    - Note if $c_1 = 0$ then the method won't converge

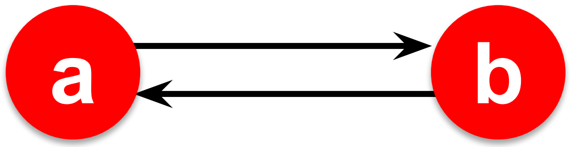
PageRank: final calculation

- For an infinitely long journey made by the surfer, the value at which converges ***p*** is simply ***the frequency at which the surfer touches every page***
- There is a close correspondence between the result of the "aimlessly surfer model" and prestigious adoption in bibliometrics: a page ***v*** has high prestige if the rate of visits is high; this happens if there are many neighbours ***u*** with high visiting rate that lead to ***v***
- Unlike index of classical prestige, the PageRank of a node is divided on its outgoing edges
- More precisely, while in social networks the prestige of starting node ***u*** is totally added to that of a target node ***v***, for the PageRank only the amount allocated on outgoing edges of ***u*** is added to the PageRank of ***v***
- In other words the PageRank can be thought of as the flow of a liquid which is preserved at each node: the amount that comes from the incoming edges is equal to what leaves the node through the outgoing arcs

PageRank: three questions

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

Does this converge?



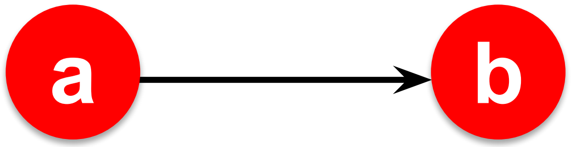
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

- **Example:**

r_a	1	0	1	0
r_b	0	1	0	1

Iteration 0, 1, 2, …

Does it converge to what we want?



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

- **Example:**

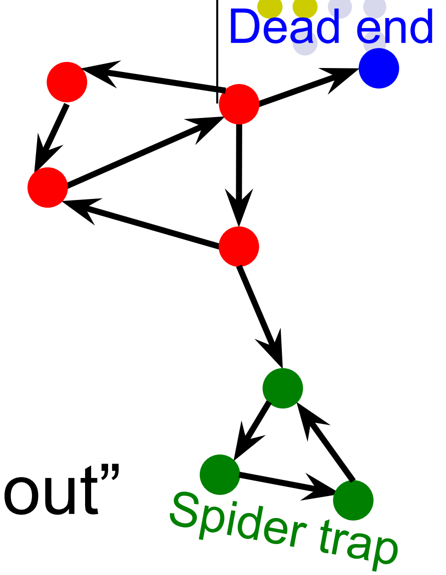
$$\begin{array}{lcccc} \mathrm{r_a} & 1 & 0 & 0 & 0 \\ \mathrm{r_b} & 0 & 1 & 0 & 0 \end{array}$$

Iteration 0, 1, 2, …

PageRank: Problems

2 problems:

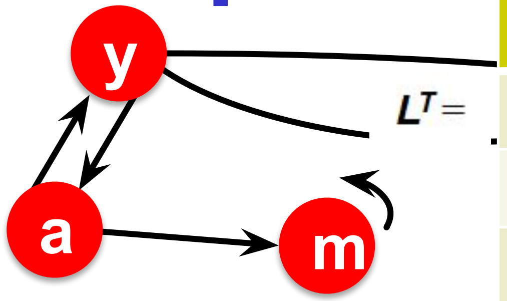
- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"



- **(2) Spider traps:** (all out-links are within the group)
  - Random walked gets "stuck" in a trap
  - And eventually spider traps absorb all importance

Problem: Spider Traps

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



$L^T =$

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

m is a spider trap

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2$$
$$r_m = r_a/2 + r_m$$

- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, …

All the PageRank score gets "trapped" in node m.

PageRank: final calculation

- The simple navigational model described above is not enough, because the Web graph is NOT strongly connected and aperiodic
- Roughly a quarter is strongly connected, there are many pages with no outgoing links, as well as direct routes leading into cycles where the path may be “trapped”
- A simple remedy is to insert fake low probability transitions everywhere. In the new graph, the navigation system at each node makes a choice between two different possibilities:
  - with probability ***d***, the surfer jumps or teleports to a web page chosen at random
  - with probability ***1-d***, the surfer jumps to a next page to the current one (a node near the current node), choosing randomly and uniformly as before an outgoing link

PageRank: final calculation

- ***d*** is a parameter, called damping factor, set in a suitable way, usually between 0.1 and 0.2 (0.15 seems to be the most likely value)

- ***The induced PageRank represents a probability distribution on the Web pages. Obviously the sum of PageRanks of all pages is 1***

- Because of the random jump, the equation $p_{i+1} = L^T p_i$ becomes

$$\mathbf{p}_{i+1} = (1-d)L^T\mathbf{p}_i + d\begin{pmatrix} 1/N & \cdots & 1/N \\ \vdots & \ddots & \vdots \\ 1/N & \cdots & 1/N \end{pmatrix}\mathbf{p}_i$$

$$= \left((1-d)L^T + \frac{d}{N}\boxed{\mathbf{1}_N}\right)\mathbf{p}_i$$

PageRank: final calculation

- In other words, the PageRank is the principal eigenvector of the matrix ***M=(1-d)L^T+(d/N) 1_N*** — rendered as $M=(1-d)L^T+(d/N)\,1_N$
- Simplifying the notation

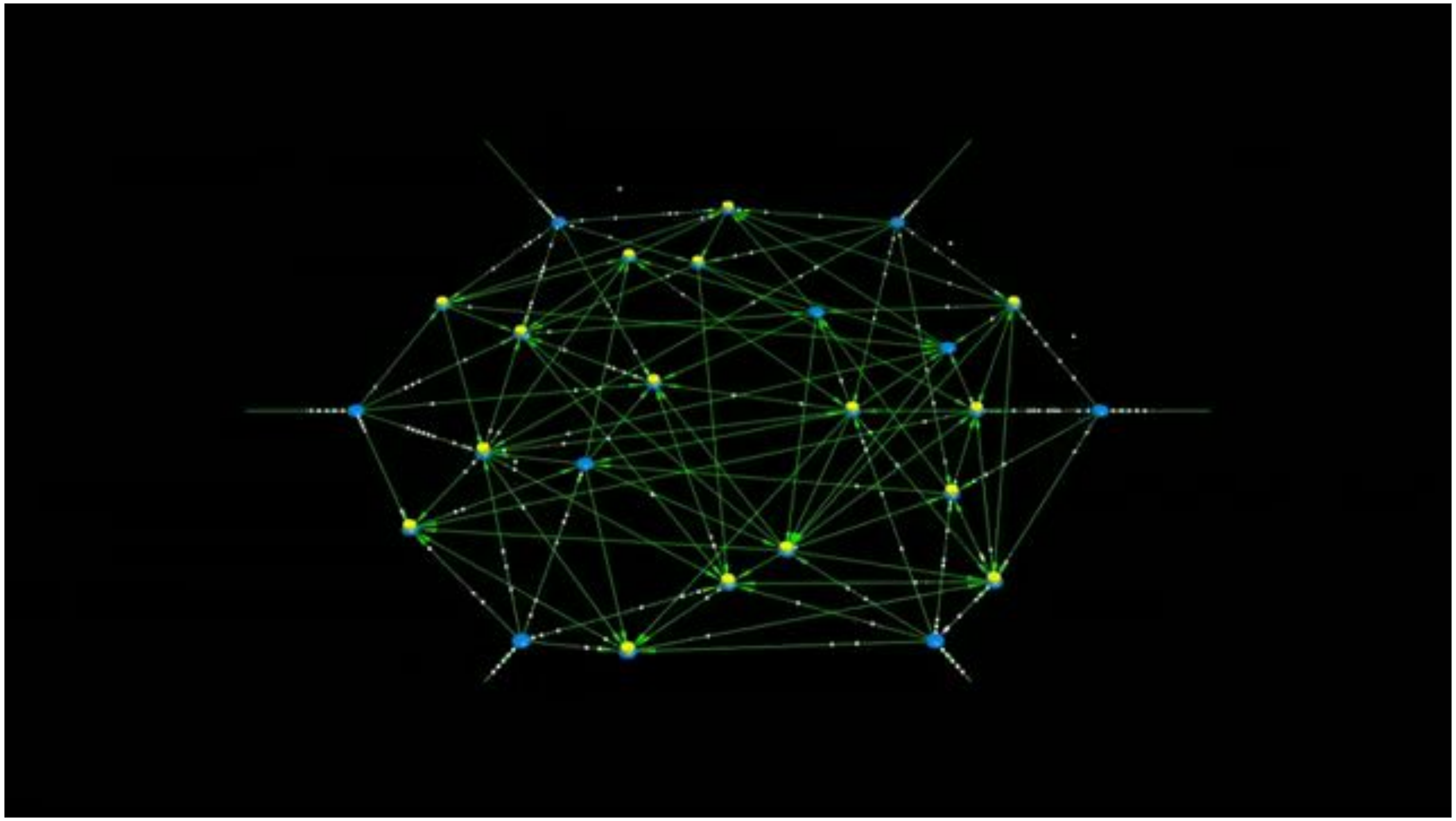
$$p_{i+1} = (1-d)L^T p_i + \frac{d}{N}(1,\ldots,1)^T$$

where ***N*** is the number of nodes in the graph.

- In other words, the value ***p[v]*** of the PageRank of a node ***v***, can be expressed as

$$p[v] = \frac{d}{N} + (1-d)\sum_{(u,v)} \frac{p[u]}{N_u}$$

PageRank in action (video)



PageRank: ensure convergence

- Given the large number of nodes and links, it is not possible to find directly a solution for the linear system of equations arising for the computation of the eigenvector of the matrix, that is $p=Mp$

- However the power iteration method can be used, starting from an initial vector p_0

- No normalization is needed, since vectors p_i are probability vectors, that is $|p_i|_1=1$

- It can perform a limited number of iterations, between 52 and 322 million pages, since it has been experimentally verified that starting from a certain iteration, even if not yet exact, the obtained values sufficiently converge to the stable ones, so as to induce the same rankings on pages

- In other words, starting from a certain iteration, continuing further does not change the order of the returned pages

- There are several ways to treat nodes with no outlink:
  - jump from them with probability one
  - remove these nodes, compute the PageRank on the residue graph , re-enter and propagate them the PageRank previously computed

PageRank: final calculation

- During the crawling phase Google uses the built graph to pre-compute and store the PageRank of the pages
- When a query is entered, an undisclosed mechanism is used to combine the PageRank with the relevance indicators of texts coming from classical Information Retrieval to determine the final order in which pages are listed
- All this is done by maintaining a speed comparable to that of classical text analysis search engines

PageRank: final calculation

- PageRank is a core classification mechanism of Google, but it is not the only one
- In fact, among other factors considered, we have:
  - Keywords
  - Sentences match
  - Proximity match
  - Hyperlinks text
- Sometimes this creates awkward situations, such as the bombing, primarily due to the text around the hyperlink (try to search with the word "failure"), which often requires a parameter adjustment based also many empirical parameters and / or intervention human
- Despite its effectiveness, Google is subject to some criticism, including:
  - PageRank, that is prestige of the pages, is defined by a single random walk not affected by the query
  - decoupling between the relevance of the page with respect to the query and the popularity of the same page and the ad hoc mechanisms used to reunification efficiently at query time

Some Problems with PageRank

- **Measures generic popularity of a page**
  - Will ignore/miss topic-specific authorities
  - **Solution:** Topic-Specific PageRank
- **Uses a single measure of importance**
  - Other models of importance
  - **Solution:** Hubs-and-Authorities
- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank