Lab 7 – HITS + PageRank + TrustRank

Deadline: +1 week

1 Motivation & Web structure mining

Web structure mining is focused on discovering a topology of the web. This structure can be represented as a directed graph. The goal of web structure mining is to generate a structural summary of this graph. This can be, e.g.,:

- finding similarities/relations between websites,
- categorizing websites: for crawling policy, for ranking purposes (search engines),
- revealing the structure of a website (navigation purposes).

2 HITS

HITS (Hyperlink Induced Topic Search; John Kleinberg, 1998) — an algorithm for ranking Web Pages according to both their content and the structure of the links. The algorithm divides pages into two groups:

- authorities -– pages being pointed by many other pages (contain important information),
- hubs -– pages linking to many authorities (show where information can be found).

2.1 Algorithm

HITS is applied on a sub-graph after a search is done on the complete graph. Firstly, the search is applied. Then, HITS analyses the structure of the links of the retrieved relevant pages.

1. Identify a set of pages being relevant according to query q (a root set R_q).
2. Extend the root set by adding:
   (a) pages which which are linked by pages from R_q,
   (b) pages which link to R_q.

The extended set is called a base set S_q.

3. Then, HITS analyzes the structure of the base set to identify hubs and authorities:

   (a) Let L be the adjacency matrix of S_q ($L_{i,j} = 1$, if page i links to page j, $L_{i,j} = 0$ otherwise).

   (b) Let $a = [a_1, a_2, ..., a_n]$ be the vector of authorities and $h = [h_1, h_2, ..., h_n]$ be the vector of hubs. These vectors contain coefficients (weights). The greater is the value of a_i (h_i), the better is the authority (hub). h and a are defined as:

   - $h_i = \sum_{j:L_{i,j}=1} a_j$,.
   - $a_i = \sum_{j:L_{j,i}=1} h_j$

4. How to find the values of a and h?

   (a) Iteratively:

      i. Initialize $a = [1, 1, ..., 1]$ and $h = [1, 1, ..., 1]$.
      ii. $h_i^{t+1} = \sum_{j:L_{i,j}=1} a_j^t$
      iii. $a_i^{t+1} = \sum_{j:L_{j,i}=1} h_j^t$
      iv. normalize a^{t+1} and h^1.

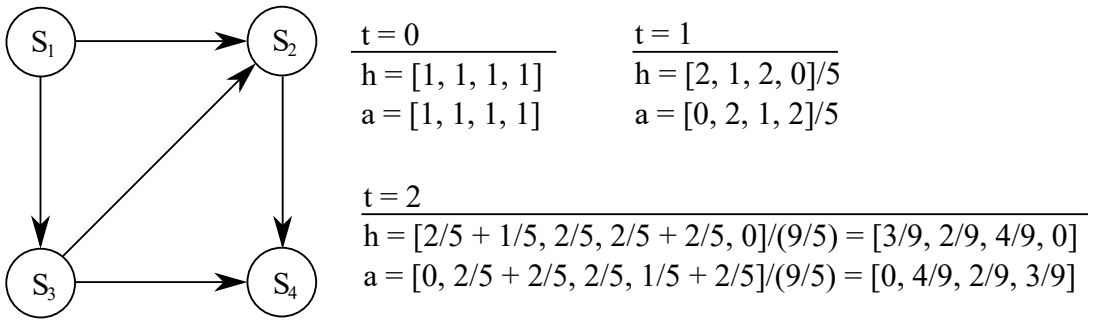


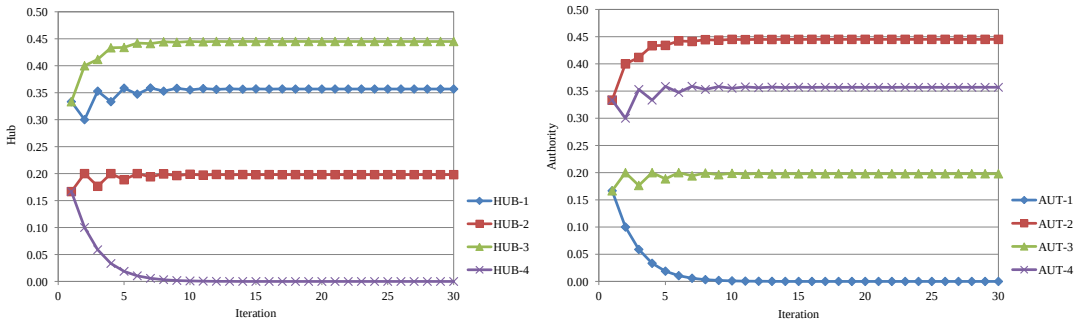
Figure 1: HITS – Iterative procedure – three iterations

   (b) Using eigenvalues:

$$\begin{cases} h^{t+1} \leftarrow La^t \\ a^{t+1} \leftarrow L^T h^t \end{cases} \rightarrow \begin{cases} h^{t+1} \leftarrow LL^T h^t \\ a^{t+1} \leftarrow L^T L a^t \end{cases} \rightarrow \begin{cases} h^{\infty} \leftarrow (1/\lambda_h) LL^T h^{\infty} \\ a^{\infty} \leftarrow (1/\lambda_h) L^T L a^{\infty}. \end{cases}$$

   The last two equations are **eigenvalue equations** of LL^T and $L^T L$. One may compute h and a by finding eigenvectors that correspond to the greatest eigenvalue of LL^T and $L^T L$, respectively:

   - **Compute hubs**



(a) Hubs

(b) Authorities

Figure 2: HITS – Iterative procedure – convergence

i. Find all λ by solving $det(LL^T - I\lambda) = 0$ (see calculator here).

ii. Select the greatest λ: $[0.1980..., 1.000..., 1.554..., 3.246...]$.

iii. Then, solve $(LL^T - I\lambda)x = 0$ for selected λ. x is then an *eigenvector*. Solution $x = [0, 1.246, 0.554, 1.000, 0.000]$.

iv. After normalization, $h = [0.356, 0.198, 0.445, 0.000]$.

- **Compute authorities**

i. Find all λ by solving $det(L^T L - I\lambda) = 0$ (see calculator here).

ii. Select the greatest λ: $[0.1980..., 1.000..., 1.554..., 3.246...]$.

iii. Then, solve $(LL^T - I\lambda)x = 0$ for selected λ. x is then an *eigenvector*. Solution $x = [0.000, 1.246, 0.554, 1.000]$.

iv. After normalization, $a = [0.000, 0.445, 0.198, 0.356]$.

(c) Or by solving the system of equations. As you may observe above: $\lambda_h = \lambda_a$. Thus the following system of equations can be solved:

$$\begin{cases} h\mu \leftarrow LL^T h \\ a\mu \leftarrow L^T L a \end{cases}$$

3 PageRank

PageRank (Page, Brin, 1998) — algorithm for ranking websites, based on the web's topology. It assumes that the popularity of the page can be measured based on how often the average user visits it. PageRank uses a "random web surfer", which clicks links on a page with some probability (e.g., equal). It represents random exploration of the web.

The importance of a page is an average importance of the pages which links to it (d_i – page i):

$$PageRank(d_i) = v_i = \sum_{d_j : d_j \to d_i} \frac{v_j}{c_j},$$

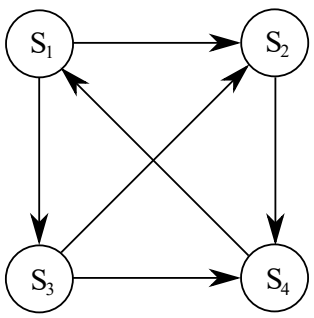
where c_j denotes the number of links the page d_j has.

Applications:

- Search engines (websites are ranked according to their PageRank; gives additional criterion based on the quality of the page, not only its content),
- predictions of Web traffic (estimation of users' visits count, server load, etc.),
- optimal crawling -– crawlers should visit important pages more frequently (important page — a page with a high PageRank value),
- website navigation,
- modeling ecosystems, protein networks.

3.1 Algorithm

1. Construct a stochastic matrix M:
   - M – matrix n x n, where n is the number of the pages,
   - $M_{i,j} = 0$, if the page j does not contain a link to the page i; $M_{i,j}$=$1/c_j$, if the page j does contain a link to page i.
2. How to find PageRank values?
   (a) Iteratively:
      i. Initialize $v = [1/n, ..., 1/n]$, where n denotes the total number of considered pages.
      ii. $v_i^{t+1} = \sum_j v_j^t M_{i,j}$
   (b) Alternatively, using eigenvalues: One may compute v by finding eigenvectors, which correspond to the greatest eigenvalue of M:
      i. Find all v by solving $det(M - I\lambda) = 0$,
      ii. Select the greatest $\lambda : [-0.340... - 0.8i, -0.340... + 0.8i, -0.319, 1]$.
      iii. Then, solve $(M - I\lambda)x = 0$ for the selected λ. x is an eigenvector. Solution: $x = [4, 3, 2, 4]$.
      iv. Normalize x: $v = [0.307, 0.230, 0.153, 0.307]$.



t = 0

v = [1/4, 1/4, 1/4, 1/4]

M = 0.0, 0.0, 0.0, 1.0
0.5, 0.0, 0.5, 0.0
0.5, 0.0, 0.0, 0.0
0.0, 1.0, 0.5, 0.0

t = 1

h = [1/4, 1/4, 1/8, 3/8] / (8/8)

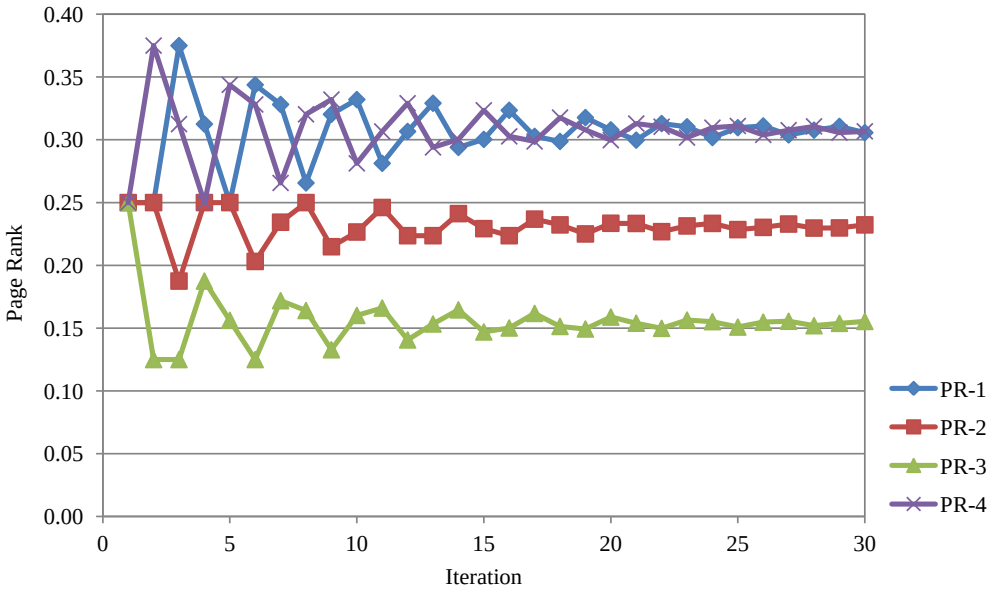
t = 2

h = [3/8, 3/16, 1/8, 5/16] / (16/16) = [6/16, 3/16, 2/16, 5/16]

Figure 3: PageRank – Iterative procedure – three iterations

(c) By solving system of equations:

$$\begin{cases} v_1 = v_4 \\ v_2 = 0.5v_1 + 0.5v_3 \\ v_3 = 0.5v_1v_4 = v_2 + 0.5v_3 \\ v_1 + v_2 + v_3 + v_4 = 1 \end{cases}$$

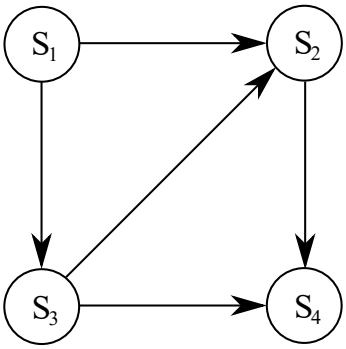


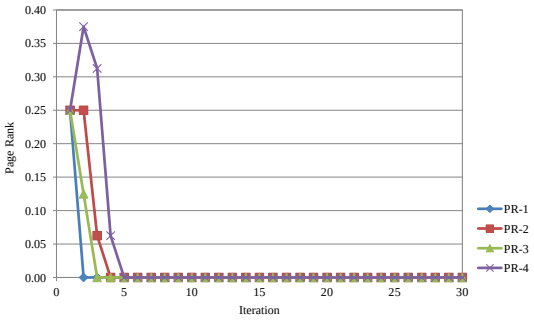
(a) PR values

Figure 4: PageRank – Iterative procedure – convergence

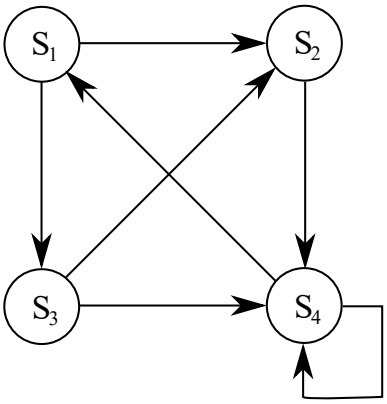
3.2 PageRank calculation problems

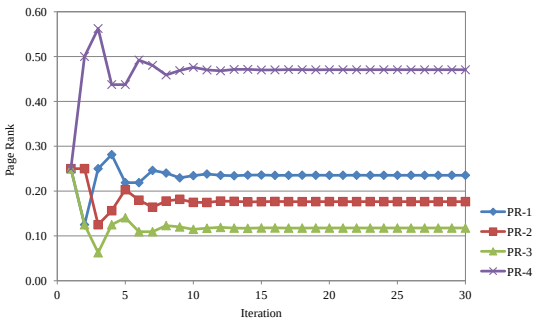
1. Dead-end:

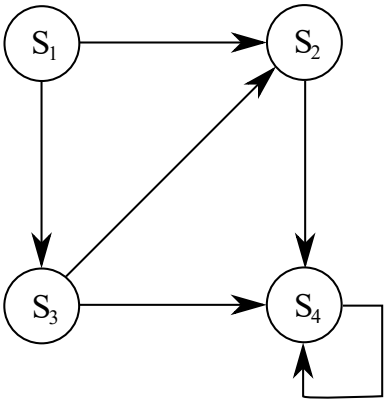


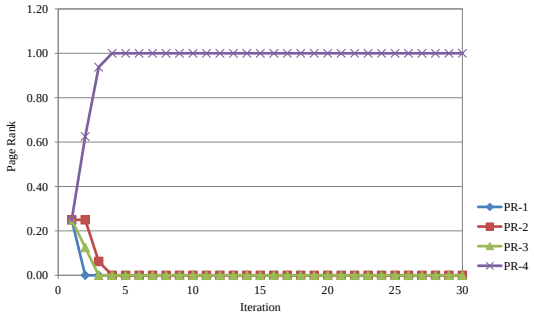


2. Spider-trap:
3. Spider-trap + dead-end:









3.3 PageRank with damping factor

Modification for dealing with "spider traps" and "dead-ends":

$$PageRank(d_i) = v_i = q + (1 - q) \sum_{j:d_j \to d_i} \frac{v_j}{c_j},$$

where q is a damping factor (usually equals to 0.15). Example for $q = 0.15$ (normalization is applied):

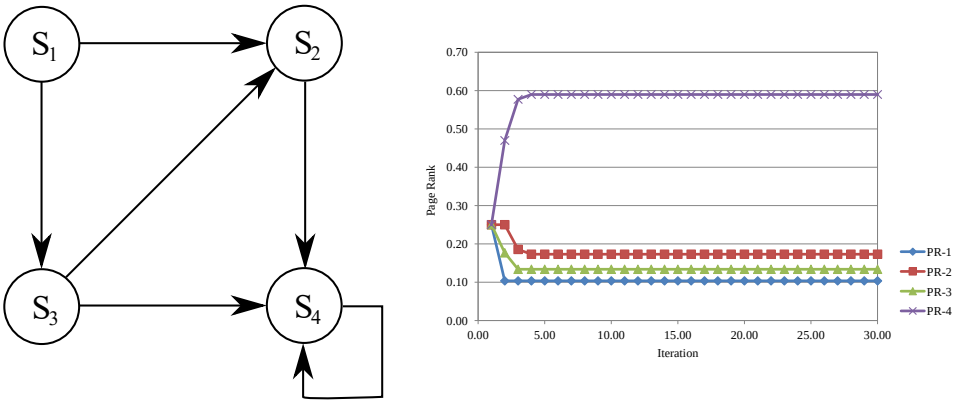


Figure 8: PageRank with damping factor

4 Link farm

A link spamming technique which goal is to maximize a page rank of some page. There exists three categories of web pages from the point of view of the spammer:

- pages that are not available – the spammer cannot add any content (comments, etc.),
- pages that are available – the spammer may add some content (comments, etc.),
- own pages. Considering pages b) and c), the spammer may insert many links to page t in order to increase its page rank.

5 TrustRank

TrustRank is a modification of PageRank whose main purpose is to handle spam (link spamming). The idea of this algorithm is that the situation, in which a "good" page links go a "bad" is very uncommon (approximate isolation):

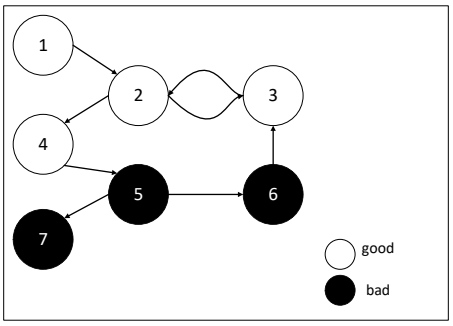


Figure 9: TrustRank

In TrustRank the trust factor is calculated for each page. Then, based on some threshold, the pages are categorized to: trusted and untrusted.

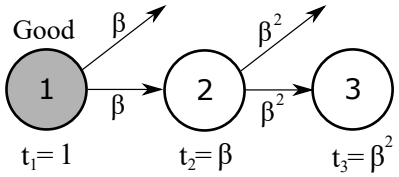
5.1 Algorithm

1. Choose a small subset of pages from the Web ("seed pages").
2. For each page from the seed determine whether it is "good" or "bad".
3. "Good" pages are marked as "trusted".
4. Initially, the trust factor of pages is computed as follows:

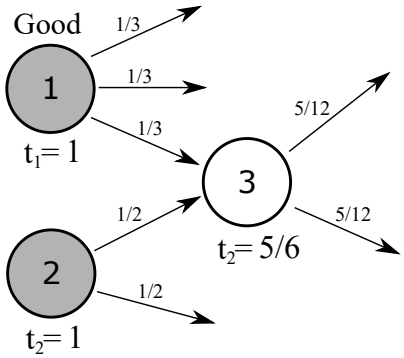
$$O(p) = \begin{cases} 0, \text{if page } p \text{ is "bad"} \\ 1, \text{if page } p \text{ is "good"} \end{cases}$$

5. Propagate trust factor values according to the structure of the network:

   - Trust factor should decrease with the distance from the trusted page (distance – number of links between pages; trust dampening):



   - Trust factor should be divided equally between all pages linked from given page (trust splitting):



6. TrustRank as a modification of PageRank:

(a) Indicate a set of "good" pages and let $d_i = 1$ if the page i is trusted, d_i=0 otherwise.

(b) Normalize $d \leftarrow d/\sum_i d_i$

(c) Set an initial vector $TR(d_i) = t_i = d_i$

(d) Apply the equation (e.g., iteratively):

$$TR(d_i) = t_i = q \cdot d + (1 - q) \sum_{d_j : dj \rightarrow d_i} \frac{t_j}{c_j}$$