

# Learning Lab Lower Saxony [L3S]

## Finding related hubs and authorities

Paul-Alexandru Chirita, Daniel Olmedilla, Prof. Wolfgang Nejdl

www.learninglab.de  
Expo Plaza 1 · D-30539 Hannover  
Tel.: +49. (0)511. 762-9731

Contact:  
chirita@learninglab.de  
olmedilla@learninglab.de

### Motivation:

Current Web search engines do not always provide the user's expected answers.

Users may want to find hubs, related to the pages they consider interesting, in order to get more information.

### Current state:

Some good ranking algorithms, but which are not combining the global importance of a page with its importance as a hub.

Few algorithms for computing related pages, almost no algorithm for computing related hubs.

### Solution:

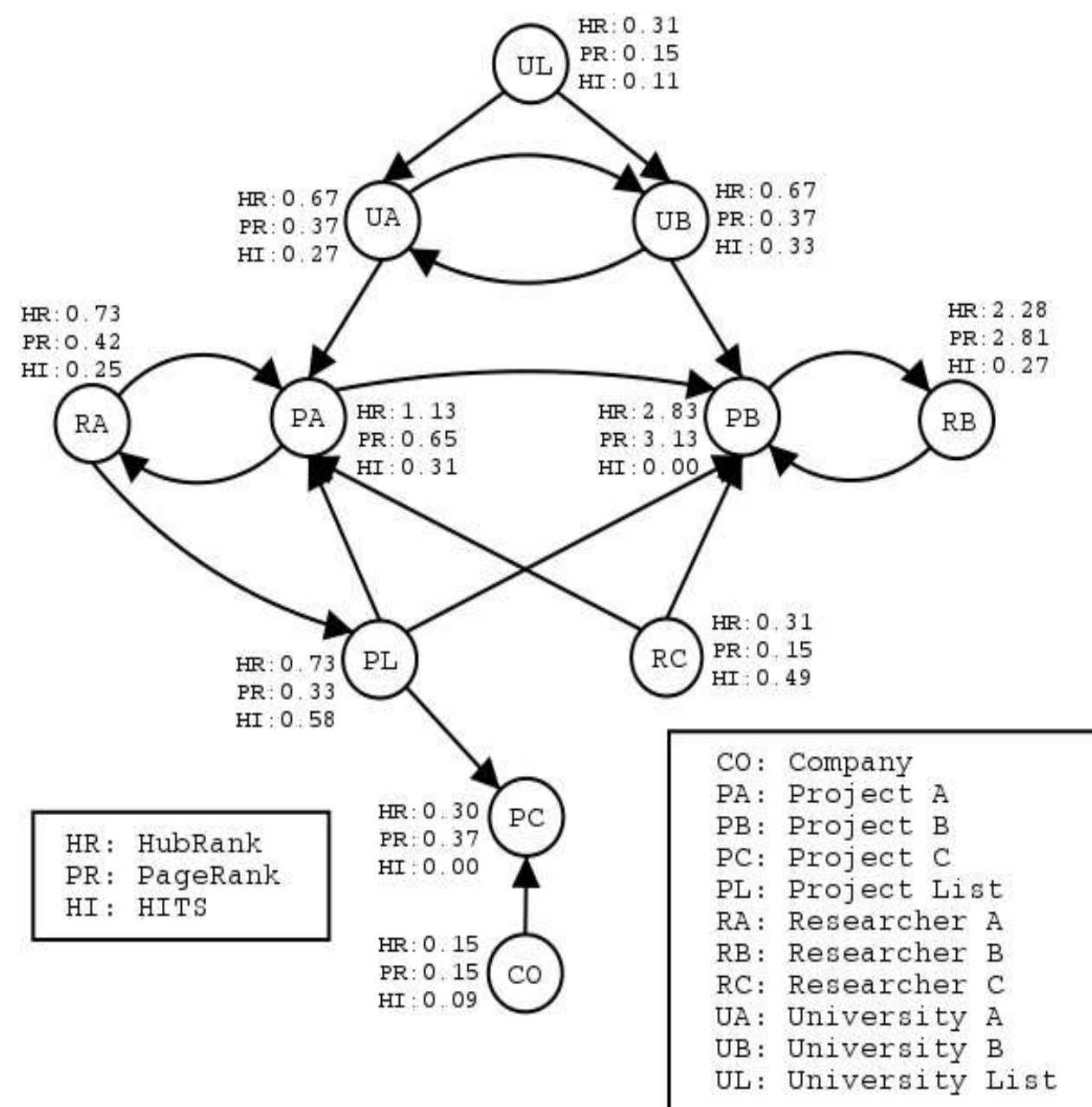
Find a new computation method to combine global page importance and hub quality measure in order to define a new ranking algorithm.

New search algorithm which finds high quality hubs connected to a given initial set of pages.

### HubRank:

PageRank has demonstrated to be one of the most important rank algorithms developed so far in order to find good authorities. However it can be slightly modified to provide better results focusing on different aspects. Our idea here is to bias the PageRank algorithm in order to rank higher pages that are likely hubs.

The algorithm for computing hub scores is depicted in the rightmost part of this row:



Let  $N$  be the total number of pages in the Web graph  
Let  $SO$  be the sum of all the out-degrees of the pages in the Web graph  
Let  $O_i$  be the out-degree of page  $i$   
Let the components of the personalization vector "e" be:

$$e_i = O_i \cdot \frac{N}{SO}$$

Apply the Google pagerank using "e" as the personalization vector:

$$x^{(n+1)} \leftarrow c \cdot A \cdot x^{(n)} + (1-c) \cdot e$$

Return  $x$

Node	Authority Quality	Hub Quality	PageRank	HITS	HubRank
Project B	+++	--	1	10	1
Researcher B	++	+/-	2	5	2
Project A	+	+	3	4	3
Project List	-	+++	8	1	4
Researcher C	---	+++	9	2	8

### HubFinder:

Kleinberg defines an algorithm to extend around an initial set of pages together with the HITS algorithm ("Kleinberg Extension" hereafter). Some previous algorithms (eg. HITS) apply this extension once and then compute hub and authority scores for the resulting pages, keeping only top-ranked results.

HubFinder is intended to be an algorithm for finding hubs related to an initial base set of pages. We define related as accessible via the link structure of the Web (following either in-going or out-going links). When searching for related hubs, one would probably need to apply the Kleinberg extension several times, because if one starts from a poor hub/authority, the representative pages might be placed quite a few links away. HubFinder improves this by decreasing the amount of pages kept after each Kleinberg extension and thus increasing computation speed.

Our idea has the following skeleton as algorithm basis:

Let  $G$  be the Base Starting Set of pages whose related hubs/authorities we are looking for  
 $G =$  Apply the Kleinberg Extension on  $G$  once (add all pages pointed to and maximum 50 pages pointing to each current page)

$G' = G$

For  $i = 1$  to  $s$  do:

$G'' =$  Apply the Kleinberg Extension on  $G'$  once

Trim  $G''$  to contain only interesting pages, which are not contained in  $G$

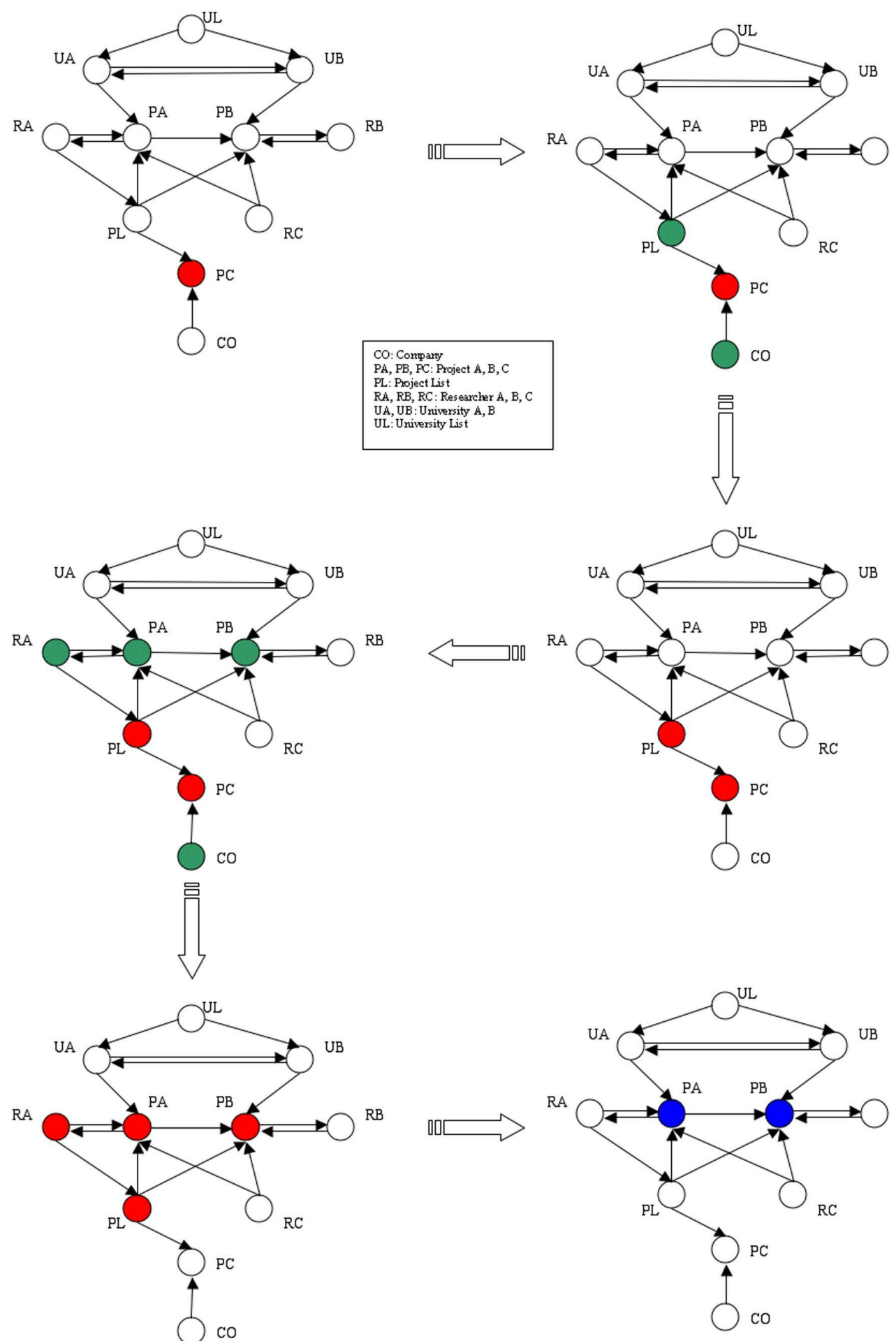
$G = G + G''$

$G' = G''$

End For

Trim  $G$  to contain only interesting pages

Return  $G$



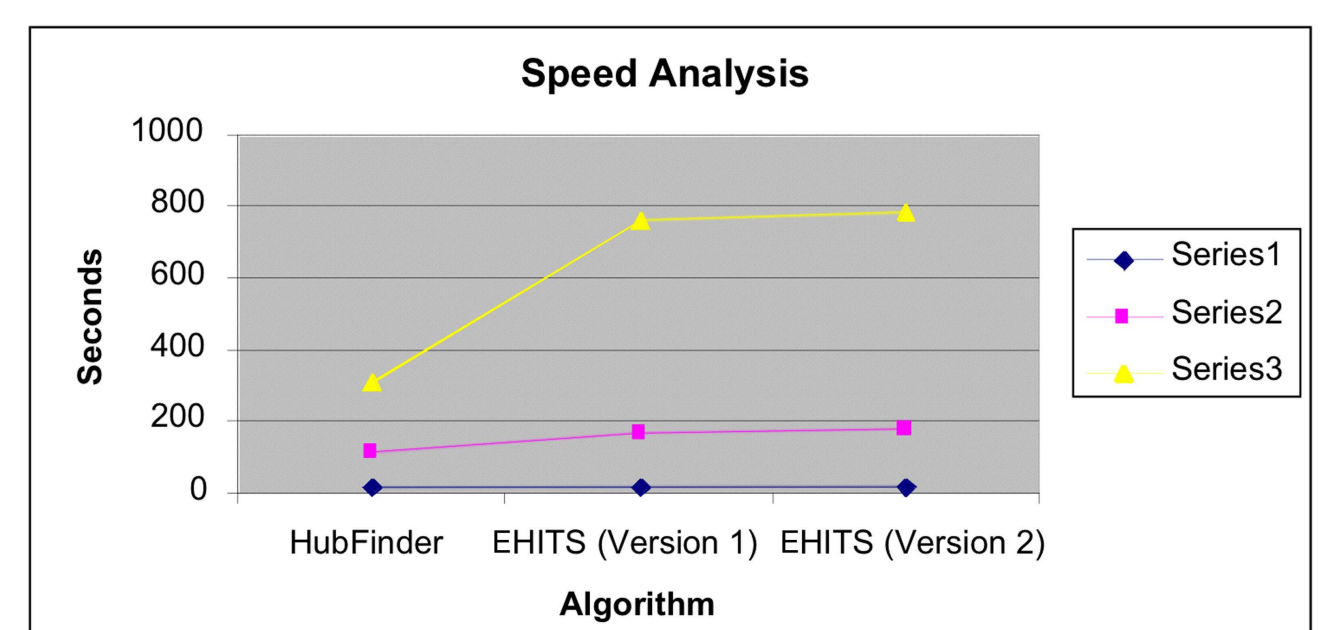
Number of pages kept after each iteration using 10000 pages crawl as input:

Step	HubFinder	EHITS (version 1)	EHITS (version 2)
1	160	180	180
2	194	254	254
3	286	937	850
4	574	2755	2763
5	833	3414	3410
Final Step	33 (4%)	34 (1%)	34 (1%)

Resulting sets similar 100% to the resulting sets obtained using the Kleinberg algorithm.

	HubFinder	EHITS (version 1)	EHITS (version 2)
Time	5.840 sec.	9.455 sec	10.383 sec

Significantly improved computation speed, while using similar amounts of memory



- Pages are more interesting when they have a bigger page rank

- The number of interesting pages adapts itself to the number of pages discovered by the algorithm. It is computed using the following formula:

$$N_{new} = \frac{100 - \lg(N_{old}) * 10}{1 + \alpha * (D - 1)} * \frac{N_{old}}{100}$$

where  $D$  represents how many steps we already performed and  $\alpha$  is a degeneration factor.

### Conclusions and further work:

We present a new algorithm for computing global page importance focusing on hub quality as well.

Users can now find faster the hubs related to their desired

pages. HubFinder is twice as faster as other existing algorithms with similar features.

We are currently extending our tests to bigger input data sets (our current set has 3 million pages)

