

Finding Related Pages Using the Link Structure of the WWW

Paul - Alexandru Chirita, Daniel Olmedilla and Wolfgang Nejdl
L3S and University of Hannover
Deutscher Pavillon Expo Plaza 1
30539 Hannover, Germany
{chirita, olmedilla, nejdl}@learninglab.de

Abstract

Most of the current algorithms for finding related pages are exclusively based on text corpora of the WWW or incorporate only authority or hub values of pages. In this paper, we present HubFinder, a new fast algorithm for finding related pages using the link structure of the Web graph as a basis. Its criterion for filtering output pages is "plug-gable", depending on the user's interests, and may vary from global page ranks to page text content, etc. We also introduce HubRank, a new ranking algorithm which gives a more complete view of page "importance" by biasing the authority measure of PageRank towards hub values of pages. Finally, we present an evaluation of these algorithms in order to prove their qualities experimentally.

1 Introduction

How to best use the link structure of the World Wide Web has been investigated quite a lot in recent years. The success of the Google PageRank algorithm [16] has inspired much of this work, but has lead also to the realization that further improvements are needed. The amount and diversity of Web pages (Google now indices about 4.3 billion pages) lead researchers to explore faster and better Web searching algorithms, to reduce the time needed by a user to find what he is searching for.

Existing Web search engines only provide restricted services for finding *related pages*. These services are generally based on text corpora of the WWW and return authorities, i.e. pages with high page rank (e.g. as computed with [16]) usually pointed to by many other pages, and especially by many hubs. While authorities certainly represent important resources (when their content/topic corresponds to user interests) we often also want to learn about the existing hubs for the topic we are interested in. Hubs are Web pages pointing to many sites in one or more domains and represent important overviews of information resources on specific

topics. Hubs therefore are often more useful as bookmarks than a set of single specific sites on one specific subject. Additionally, they have many other utilities. Let us imagine a user who wants to buy a car but he does not know which one. After searching for a "car store in Madrid" in a search engine, he obtains a set of pages with the most "important" car stores in Madrid. However, this set is by far not complete. It is only the set of the most "important" pages according to search engine's measure of importance. A good hub would provide in this case a complete list of all the car stores registered in Madrid, helping our user to decide which car he wants and where to buy it.

The algorithms presented in this paper help users reduce the time spent searching for related pages and introduce a more complete view of page "importance", which considers not only authority, but also hub values.

Our main contribution is a new, fast and flexible algorithm (*HubFinder*) for finding hubs (or authorities) related to a given initial set of pages. A related page of any type is one that address the same topic as the original page [7] (e.g. both are newspapers). Similar to [11], we use the link structure of the World Wide Web as input. If there is a link from page v to page w , then the author of v recommends page w , and the two pages are usually related. However, we will also propose a refinement for our algorithm based on the content analysis solutions proposed in [1]. Furthermore, we have developed a modified version of the PageRank algorithm (which we called *HubRank*), meant to bias the original PageRank values combining both the authority and hub value of each Web page. We use this algorithm to filter the related pages we find. Finally, we have performed several experiments in order to evaluate the quality of our two new algorithms.

The paper is organized as follows. We will start by shortly discussing work related to our research in section 2. The next two sections describe our new algorithms, HubFinder and HubRank. Section 5 shows first experimental results, and section 6 concludes with discussion of further work.

2 Previous Work

2.1 Web Graph Notation

In the paper we use the following notation: $G = (V, E)$ represents the *Web graph*, where V is the set of all Web pages and E is the set of directed edges $\langle p, q \rangle$. E contains an edge $\langle p, q \rangle$ iff a page p links to page q . $I(p)$ denotes the set of in-neighbors (pages pointing to p) and $O(p)$ the set of out-neighbors (pages pointed to by p). For each in-neighbor we use $I_i(p)$ ($1 \leq i \leq |I(p)|$), the same applies to each out-neighbor. We refer $v(p)$ to denote the p -th component of \mathbf{v} . We will typeset vectors in boldface and scalars (e.g., $v(p)$) in normal font. Let A be the adjacency matrix corresponding to the Web graph G with $A_{ij} = \frac{1}{|O(j)|}$ if page j links to page i and $A_{ij} = 0$ otherwise.

2.2 Ranking Web Pages

PageRank [16] is an algorithm for computing a Web page score based on the graph inferred from the link structure of the Web. Its motivating idea is that pages with many backlinks are more important than pages with only a few backlinks. As this simple definition would allow a malicious user to easily increase the "importance" of his page simply by creating lots of pages pointing to it, PageRank uses the following recursive description: "a page has high rank if the sum of the ranks of its backlinks is high". Stated another way, the vector **PR** of page ranks is the eigenvector of A corresponding to its dominant eigenvalue.

Given a Web page p , the PageRank formula is:

$$PR(p) = (1 - c) \sum_{q \in O_p} \frac{PR(q)}{\|O(q)\|} + cE(p) \quad (1)$$

The dumping factor $c < 1$ (usually 0.15) is needed to guarantee convergence (A by itself is not irreducible, i.e. G is not strongly connected) and to limit the effect of rank sinks [3]. Intuitively, a random surfer will follow an outgoing link from the current page with probability $(1 - c)$ and will get bored and select a new page with probability c .

HITS [12] computes two different scores for each page of a Web community (or of the whole WWW): a *hub* score and an *authority* score. The algorithm can be split into the following steps:

1. *Select a starting set of pages.* The algorithm is usually focused on a subgraph of the Web, which contains pages on a given topic. This base set can be obtained by sending a specific query to a search engine and selecting the most important k results.
2. *Extend the starting set of pages,* because some of the authoritative sources might not be in the above set.

3. *Calculate the scores.* Compute an authority measure and a hub measure for each page.

The second step is described in section 2.3. Here we will describe the process of computing Hubs and Authorities. For any page we associate two different values: a non-negative *authority weight* a^p and a nonnegative *hub weight* h^p . A higher authority or hub weight of a page indicates a higher usefulness of this page as authority or hub page. The computation consists of applying the following two operations iteratively, until convergence is reached:

Operation ϱ :

$$a^p \leftarrow \sum_{q:(q,p) \in E} h^q \quad (2)$$

Operation γ :

$$h^p \leftarrow \sum_{q:(p,q) \in E} a^q \quad (3)$$

After each step the values need be normalized, such that their squares sum to 1, i.e.: $\sum_{p \in S_\sigma} (a^p)^2 = 1$ and $\sum_{p \in S_\sigma} (h^p)^2 = 1$.

Other algorithms. HITS suffers from the community effect, i.e. many pages from the same site may have very similar scores. This problem is solved in Randomized HITS [15] where HITS' original formulas are combined with the power iteration of PageRank. [8] improves global Web page scores using site hierarchy information. It is orthogonal to HubRank, which we will discuss in section 3, where we address page location directly in the initial ranking model. Similarly, [2] presents several design guidelines meant to boost the PageRank of a page/community. Finally, an approach closer to ours could be found in [14] where the authors enhance PageRank by calculating random surfer's topic of interest and transition probabilities based on its surfing history.

2.3 Finding Related Pages

In order to find a good set of related pages, we have to take into account the following aspects: it has to be small, rich in relevant pages and it should contain many strong authorities. [12] extracts the top results of a query sent to a search engine and builds a focused sub-graph of the WWW around them. The reason to do this is that if we already have some important authorities, it is highly probable that some of these pages will point to or will be pointed to by other high authorities. The author extends the base set by adding all pages pointed to and at most d pages pointing to each page of it. We call this operation *Kleinberg extension*. In [12] the initial set is extended only once, but the author focuses on computing Hub and Authority scores, while we are focusing on finding related pages or hubs. Moreover, one can extend a set several times, by iteratively applying

the Kleinberg extension on the resulting set of the previous iteration. We will use this operation to extend our initial set.

[1] discusses an approach related to ours, which finds related pages based on the HITS algorithm plus several improvements. The most important one is the computation of relevance weights for nodes relative to a starting node, using the concatenation of the first 1000 words from the starting document (addressed as "query") and cosine normalization for weighting the query and the other documents.

The Companion algorithm [7] finds related pages by building a weighted graph around the starting page and then applying a modified version of HITS. It is designed for applying only one Kleinberg extension, whereas in HubFinder we can iterate as much as desired (thus improving the quality of results) and we use the original Web graph.

SimRank [11] analyzes similarity between graph nodes by constructing a node-pairs graph G^2 with SimRank similarity scores. The scores are based on the theory that two objects are similar when they are referenced by similar objects and reflect the node similarities only at a pair level.

[10] is similar to HubFinder in the sense that (almost) identical pages are searched in the neighborhood of each starting page. It first builds content signatures of the initial documents using the Random Projection algorithm and then propagates these signatures through the hyperlink structure in order to also include context information into the analysis. Its output is a set of (near-)replicas documents on the Web, whereas HubFinder outputs *similar* pages, whose content might be relatively different from the starting set (but covering the same topic).

3 HubRank

PageRank has demonstrated to be a successful ranking algorithm although it focuses on authority values exclusively. On the other hand, HITS computes hub ranks according only to hub values. However, if a user searches for "travel agency", she would probably be interested in a high-quality hub page with a list of all the travel agencies, a result which can be provided by neither one of these algorithms (PageRank output would not be a hub and HITS hub output would not consider authority value). HubRank addresses this problem by combining both approaches into a single score which biases PageRank [16] towards hubs.

The motivating observation is that a page pointing to a good hub is a candidate to have a high hub rank as well. Many times we encounter pages (perhaps good authorities) with only a few out-going links, but towards very important hubs. We consider such pages more important than the hubs themselves, because while a hub can cover lots of topics, such a page will usually contain information about the content addressed by the hubs it is pointing to, about the value of their content (e.g. author opinions), etc.

To achieve these hub scores, we modify the PageRank personalization vector (\mathbf{E}) to consider the out-degree/in-degree of the pages. More intuitively, the random surfer will always prefer pages with a big out-degree when it gets bored. This way, the global importance of the pages will play an important role in defining general scores, as the random surfer will follow the out-going links with a higher probability than the random ones, and on the other hand, the out-degree of pages will always be considered. However, the dumping factor should be greater than 0.15, i.e. the random surfer should prefer the random pages more often. We set its value to 0.25, although tests with values of 0.30, 0.25, 0.20 and 0.15 showed only small differences, even though the personalization vector contained quite different values (e.g. some pages had an out-degree of 300, while others had an out-degree of 3). In PageRank, the vector \mathbf{E} is a uniform distribution with $\frac{1}{NP}$ in each entry (where NP is the total number of pages). We set the value of each entry i of \mathbf{E} to $E_i = \frac{|O(i)|}{|O|} \frac{NP}{|O|}$ where $|O|$ is the summation of the out-going links over the whole Web graph.

Analogously, authority scores can be computed setting the components of the personalization vector to $E_i = \frac{|I(i)|}{|I|} \frac{NP}{|I|}$, where $|I|$ is the summation of all the in-degrees of the pages in the Web. However, when computing authority values, one might use a different matrix than the one used in PageRank (the row-out-going links matrix of the Web graph normalized on columns), depending on how much importance needs to be given to hub values and how much to authority values (e.g. the transposed row-out-going links matrix normalized on rows, as in [15]).

4 HubFinder

Introduction. HubFinder is an algorithm for finding hubs related to an initial base set of Web pages. There are many ways in which *related* can be defined. A first approach might be based on collaborative filtering algorithms, like for example in [13], where documents (i.e. Web pages in our case) are classified based on word occurrences. In [9] several other similarity measures are presented, including Pearson correlation, Spearman correlation and cosine similarity. The authors use them to compute similarity between users, but one could also use the vectors of words occurrences in documents to compute similarities between them. Such approaches provide good results, but need too much time to compute when working with huge sets of pages like the WWW.

In [11] the similarity between graph nodes is treated at the link level. In HubFinder we define *related* similarly, i.e. using only link information as input. Two pages are related if one is accessible from the other via the link structure of the Web graph (following either in-going or out-going links). The distance (the number of links followed)

between two such pages is usually less than six. According to our experiments, if the distance is bigger, the link information becomes insufficient to say that pages are similar in content with a high enough probability (a similar conclusion can be found in [4]). This approach resembles some focused crawler policies such as considering an outlink v of a relevant page u also likely to be relevant [5].

Base Algorithm. Previous algorithms (such as those from [12] or [1]) focus on applying the Kleinberg extension once and then computing hub and authority scores for the resulting pages, in several ways. When searching for related hubs (authorities), one often needs to build a larger set of pages before having found enough representative pages (good hubs/authorities). Such a set can only be built by applying the Kleinberg extension several times, because if one starts from a poor hub/authority, the representative pages related to it might be placed quite a few links away. Note that [12] starts from top-ranked pages, which is not always the case for HubFinder. In this scenario, the size of the resulting set might become very big, even if the maximum number of links followed is 6, as we suggested above. In some cases, we discover up to 500,000 pages around a single page or even more and the computation becomes too slow. HubFinder tackles this problem by trimming the pages discovered after each Kleinberg extension. It has the following algorithm as a basis:

```

Let  $\Gamma$  be the Base Starting Set of pages whose related hubs
(authorities) we are looking for
 $\Gamma \leftarrow$  Apply the Kleinberg Extension on  $\Gamma$ 
 $\Gamma' \leftarrow \Gamma$ 
For  $i = 1$  to  $\sigma$  do:
   $\Gamma'' \leftarrow$  Apply the Kleinberg Extension on  $\Gamma'$ 
  Trim  $\Gamma''$  to contain only interesting pages,
  which are not contained in  $\Gamma$ 
   $\Gamma \leftarrow \Gamma + \Gamma''$ 
   $\Gamma' \leftarrow \Gamma''$ 
End For
Trim  $\Gamma$  to contain as many interesting pages as desired
Return  $\Gamma$ 

```

The first particular aspect of this algorithm is to determine *which are the interesting pages*. This depends on the approach we take. We could calculate the PageRank vector for the small Web graph created after applying the Kleinberg extensions, or similarly, apply HITS on it. However, a page with high global PageRank is more important (globally) than a page with high local PageRank and small global PageRank. Also, even when thinking that a locally high rated page might be closer to user's preferences than a globally high rated one, we may argue that if the latter is accessible from the user's starting set, then it is considered at least partially important by the user. Therefore, we use global ranking scores, but in principle they could be local as well. Another variable that needs to be defined is *how many pages we actually keep after a trimming step*. The specific number should be a percent from the existing pages

(e.g. the top 5% of the interesting ones). A possible formula for it is depicted and then discussed below.

$$N_{new} = f(N_{old}) * \frac{N_{old}}{100} = \frac{100 - \lg(N_{old}) * 10}{1 + \alpha * (D - 1)} * \frac{N_{old}}{100} \quad (4)$$

The variables have the following meaning:

- N_{old} is the number of pages before the trimming step
- N_{new} is the number of pages after the trimming step
- D represents the current distance from the starting set (how many Kleinberg extensions we applied, or, more specific, the maximum distance of links from any page of the starting set to any page of the current set)
- α is a constant called degeneration factor

The main requirements in designing the formula are to consider the number of Kleinberg extensions applied and to generate smaller values for bigger sets and bigger values for smaller sets. The first criterion is tackled by D in the denominator. A hub five links away from the base set is not as likely to be related to it as a hub which is only one link away. The further we go exploring the link structure of the Web, the more important a page has to be in order to be kept. Moreover, α is a constant degeneration factor which allows us to set an importance amount for each distance unit. To accomplish the latter criterion, we introduced the decimal logarithm of the old number of pages in the numerator.

Extensions. When we look for hubs, HubFinder can be further filtered after each step in order to contain only pages with an *acceptable out-degree*. We defined acceptable as $Min(2 + Distance, 10)$. The further a page is from the base page considered, the bigger an out-degree it needs in order to be kept. This approach decreases the number of explored pages by several orders of magnitude, as we will see in section 5. The output set of pages is a subset of the output set we obtain using non-filtered HubFinder.

Finally, we describe a possible improvement of HubFinder based on [1], which we are currently implementing. It improves the quality of the results by eliminating the non-relevant nodes (i.e. pages not supposed to be related to user's query) from the resulting set using content analysis.

Let R be the set of results from applying HubFinder to a page p . First, compute the vectors of the first 1000 words from each output document D_j and from p . Then the similarity between p and each D_j can be expressed as in [1]:

$$similarity(p, D_j) = \frac{\sum_{i=1}^r w_{ip} * w_{ij}}{\sqrt{\sum_{i=1}^r w_{ip}^2 * \sum_{i=1}^r w_{ij}^2}} \quad (5)$$

where $w_{ip} = TF_{ip} * IDF(i)$ and $w_{ij} = TF_{ij} * IDF(i)$. TF_{ix} is the frequency of term i in document x and $IDF(i)$ is an estimate of the inverse document frequency of term i on the World Wide Web.

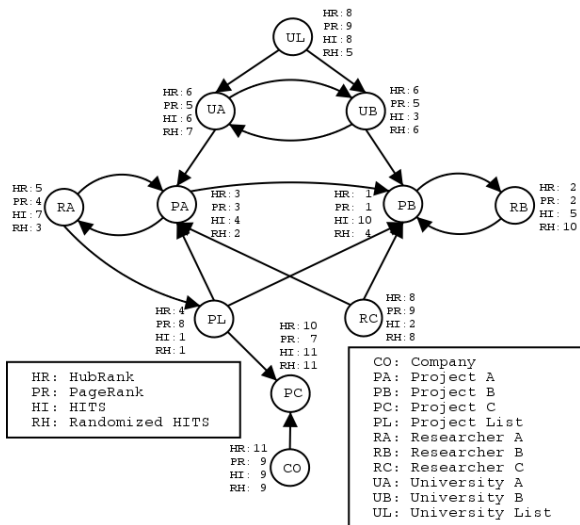


Figure 1. Small graph ranking comparison

Only those documents from the final set with a similarity value greater than a certain threshold are then returned.

5 Experimental Results

5.1 HubRank

Let us first discuss HubRank on a miniature Web graph, to demonstrate the main effects of the algorithm. Figure 1 summarizes the results of applying HubRank, PageRank, HITS and Randomized HITS on it.

A good example of the utility of HubRank is the *Project List*. As we can see, HITS and Randomized HITS rank it in a top position (first or second), as it is the best hub. PageRank, on the other hand, considers the page a poor authority and does not give much importance to it, although it might be very useful. Finally, HubRank, considering both aspects (the fact that the *Project List* is a very good hub and the fact that it is a poor authority), ranks it on place 4. Similarly, it also gives a slightly higher rank to *Researcher C* (place 8 out of 11 in HubRank, while it is the last in PageRank).

High authorities with no hub value are likely to have a decreased score. *Project B* is a very good authority and therefore still the first in HubRank, but its score is about 10% lower than the one computed with PageRank and this would result in a small rank decrease with bigger graphs. *Project C* has no hub value at all, which is materialized in HubRank by a rank decrease from 7 to 10 (out of 11) - which we intended. Usually such a decrease is smaller than the increase produced by a page being a good hub and a poor authority, because this latter type of page is more common and should be partially promoted (i.e. if one creates a new but very good hub, we try to promote him/her

URL	PR	HR	RH	HI	HSc
http://www.positiongeek.com/...	1	1992	1732	*	8.7e-32
http://www.cruisehawaii.com/...	2	1995	1763	*	8.7e-22
http://www.rom.on.ca	3	344	*	*	4.3e-12
http://www.ashmol.ox.ac.uk/g...	4	193	*	*	1.9e-17
http://www.sobotainfo.com	5	1388	*	*	9.8e-19
http://www.edinburgh.gov.uk	6	1077	23931	*	1.4e-19
http://www.commondreams.org	7	164	3199	*	1.5e-11
http://phorum.org	11	1	*	*	5.4e-10
http://www.internet.com	75	2	*	*	5.1e-8
http://www.internet.com/corp...	80	3	*	*	5.2e-8
http://www.internet.com/corp...	83	4	*	*	4.9e-8
http://www.bls.gov	67	5	34853	*	9.1e-12
http://www.fedstats.gov	15	6	*	*	1.9e-11
http://openwiki.com/ow.asp?T...	293	7	*	13248	3.5e-6
http://www.musicgearreview.c...	1710	264	1	*	8.9e-8
http://www.unep-wcmc.org/sit...	*	9424	2	*	8.7e-21
http://www.amnestyusa.org/al...	17408	7121	3	*	5.5e-15
http://www.musicgearreview.c...	1709	263	4	34499	3.4e-7
http://www.barco.com/HomeCin...	*	15438	5	*	2.0e-34
http://www.adobe.com/support/...	*	9598	6	*	5.3e-13
http://www.scenicrentals.com/...	20	133	7	*	1.2e-12
http://www.lotus.com/services...	*	*	*	1 - 7	*

Table 1. Ranking comparison for a big crawl

slightly faster than in PageRank).

Generally then, we can summarize the outcome of HubRank as follows. A strong authority will always have a good rank, but it will be top ranked only if it is at least an average hub (otherwise it might drop some ranks, compared to PageRank). An average authority will have a rank very dependent on its hub value. It could raise or drop quite a lot from its PageRank place. Finally, a poor authority will have a low rank if it has a low or average hub value, but it may get a significant raise when it has high hub value.

HITS and Randomized HITS separate hub values and authority values, which is not useful for us. For example, HITS ranks as second the node *Researcher C* that has no authority at all. On the other hand, PageRank is not considering hub importance at all, which could generate rank drops for important pages, and therefore loss of information. HubRank biases the best authorities of the Web graph according to their value as hubs, so it is more accurate than the other algorithms presented on this example.

The qualities of HubRank can be nicely discussed on small Web graphs as above, but we think they also materialize in the ranks computed for our 3 million pages crawl, in table 1, where we again compare several ranking algorithms¹.

¹A * means that the page was not in the top 50,000 pages; we had to trim long URLs in order to keep the table small

Step	HubFinder					EHITS(v1)	EHITS(v2)		
	Discovered Pages	New pages	New and Interesting Pages	Percent of Pages Kept	Total pages	Total pages	Discovered pages	New pages	Total pages
1	125	125	125	100%	128	128	125	125	128
2	180	52	43	82.692%	171	180	180	52	180
3	143	78	39	50%	210	254	148	74	254
4	536	481	159	33.056%	369	937	679	596	850
5	1913	1718	415	24.15%	784	2755	2578	1913	2763

Table 2. Performance of trimming formula

Criterion	HubFinder	HubFinder with Out-degree Filter	EHITS(v1)	EHITS(v2)
Total time needed (without including I/O time)	06 min 15 sec	05 min 52 sec	60 min 27 sec	40 min 53 sec
Average time per starting page	13 sec	12 sec	2 min	1 min 22 sec
Approximate number of pages explored	102,000	18,000	905,000	760,000 - no page was explored twice
Avg. nr. of pages explored per starting page	3,400	600	30,166	25,333
Number of pages kept	1181 (H_1)	989 (H_2)	1219 (H_3)	1121 (H_4)

Table 3. Performance of HubFinder

The ratio hub value / authority value is increasing from left to right. HubRank (HR) improves PageRank (PR) by increasing the score of hubs with average (or low) authority value. This can be immediately proved using pages' HITS hub score (HSc), which is generally much higher for top HubRank pages than for top PageRank ones. However, no matter how good it is as a hub, a page cannot be top ranked in HubRank if its authority value is very low. This is the most important difference of HubRank compared to HITS (HI) and Randomized HITS (RH), where such a page could be ranked high as a hub, while in HubRank it would only have a much higher rank than in PageRank.

The top pages of PageRank and HubRank are not top ranked in Randomized HITS, which can be explained by the fact that they are mainly authorities. On the other hand, top Randomized HITS pages are also present in the top results of HubRank, which confirms the latter's preference for hubs. Therefore, while PageRank computes the best pages from one point of view (that of authority values) and Randomized HITS computes the best pages from another point of view (that of hub values), HubRank computes a hybrid value, promoting both high authorities and "high hubs which are at least average authorities". Finally, we should add that it is not sufficient to have many out-going links in order to be ranked higher in HubRank, as hub values reinforce themselves through the fixed point iterations.

HubRank could be also used for computing hub scores, but as it starts from a different point than the algorithms for computing hub and authority values (e.g. Randomized HITS or HITS), it should be rather used as a PageRank like value, with an additional focus on hub qualities.

5.2 HubFinder

In [12] the author extends an initial set of pages using the Kleinberg extension and then computes HITS scores for the newly obtained pages in order to filter them. Both the pages in the initial set and those in the extended one are answers to a user query and should therefore be related. We tested HubFinder against two extensions of this algorithm. In the first one (which we called EHITS(v1) – Extended HITS), a starting set of pages is extended several times using the Kleinberg extension and in the end a trimming step is performed. The second version of the algorithm differs from the first one only when applying the Kleinberg extension. In EHITS(v1), at each step we apply the Kleinberg extension on all nodes of the graph and some pages are explored several times. The problem is solved in EHITS(v2), where we apply the Kleinberg extension only on the pages discovered at the previous iteration (graph leaves) and combine the result with the already existing set. We had to extend HITS with more Kleinberg extensions, not only because our main goal is to find related pages, but also because while in the original algorithm the base set contains only top ranked pages, we want to have any kind of page in this starting set (and the high authorities/hubs might be further links away).

In table 2 we present how our trimming formula performs in practice. The reader will notice later that the number of pages explored by HubFinder is far less than the number of pages explored using the EHITS algorithm, which is actually the main reason for its speed improvement. We observe that the percentage of pages decreases faster at the initial steps (first two or three steps) and then slower, depending more on the set size as we explore further. In the end, we perform an additional trimming step, in order to

adjust the size of the output set to the size we want.

The next experiment was performed on the same crawl, using a set of 30 pages as starting set, a degeneration factor equal to 1.500 and a maximum radius of 3 around the starting set. Its results are summarized in table 3.

Before analyzing the test, we should add that all algorithms use similar amounts of main memory. The time used by HubFinder here is significantly better than the time needed by EHITS, though it depends very much on the size of the Web crawl and the starting set of pages. The most expensive computation is that of applying the Kleinberg extension and therefore it is the one making the difference (HubFinder explored fewer pages than EHITS). Finally, the resulting sets of pages are a bit different in size, but with very similar content. Using the notation from our figure, we have $H_2 \subset H_4 \subset H_1 \subset H_3$. Theoretically, the sets might differ by a few pages, but we did not encounter this in our experiments.

HubFinder allows personalization by means of different criteria “plugged into” it. It can thus be used with any filtering criterion, depending on the pages one wants to obtain as result. One could for example use HubRank or PageRank to obtain globally appreciated pages (i.e. authorities). HubRank is also the best criterion when computing input sets for the Personalized PageRank algorithm [6]. When used with Randomized HITS, HubFinder produces more pages before the final trimming step. Further experiments are needed to analyze in depth which algorithm is best for specific situations.

6 Conclusions and further work

Most current algorithms for finding related pages are exclusively based on text corpora of the WWW. Even those which incorporate page ranks do this by means of rankings which consider either only authority or hub value.

In this paper, we presented HubFinder, a new algorithm for finding related pages using the link structure of the Web graph as a basis. Its criterion for filtering output pages is flexible, depending on the user’s interests, and also has different levels: (1) “pluggable” global page ranks, (2) page out-degree, and/or (3) text content of each output page. Moreover, in our experiments it proved to be faster than the other algorithms we implemented, but provided results of similar quality. We also presented HubRank, a new ranking algorithm which gives a more complete view of page “importance” by biasing the authority measure of PageRank [16] towards hub values of pages.

We are currently integrating these algorithms into a personalized ranking platform for search engines [6], as they proved very helpful in automating the generation of the input data set needed for the personalization algorithm.

References

- [1] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 104–111, Melbourne, AU, 1998.
- [2] M. Bianchini, M. Gori, and F. Scarselli. Pagerank: A circuitual analysis. In *In Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, 2003.
- [3] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.
- [4] S. Chakrabarti, M. Joshi, K. Punera, and D. Pennock. The structure of broad topics on the web. In *Proc. 11th International World Wide Web Conference*. ACM Press, 2002.
- [5] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *Proceedings of the 8th International World Wide Web Conference*, 1999.
- [6] P.-A. Chirita, D. Olmedilla, and W. Nejdl. Pros: A personalized ranking platform for web search. Technical report, L3S and University of Hannover, Feb 2004.
- [7] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1467–1479, 1999.
- [8] C. Ding and C.-H. Chi. A generalized site ranking model for web ir. In *In Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, 2003.
- [9] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, 1999.
- [10] E. D. Iorio, M. Diligenti, M. Gori, M. Maggini, and A. Pucci. Detecting near-replicas on the web by content and hyperlink analysis. In *In Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, 2003.
- [11] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.
- [12] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [13] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, 1998.
- [14] B. L. Narayan, C. A. Murthy, and S. K. Pal. Topic continuity for web document categorization and ranking. In *In Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, 2003.
- [15] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *Proc. 24th Annual Intl. ACM SIGIR Conference*. ACM, 2001.
- [16] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.