PROS: A Personalized Ranking Platform for Web Search

Paul - Alexandru Chirita[1], Daniel Olmedilla[1], and Wolfgang Nejdl[1]

L3S and University of Hannover
Deutscher Pavillon Expo Plaza 1
30539 Hannover, Germany
{chirita, olmedilla, nejdl}@learninglab.de

Abstract. Current search engines rely on centralized page ranking algorithms which compute page rank values as single (global) values for each Web page. Recent work on topic-sensitive PageRank [6] and personalized PageRank [8] has explored how to extend PageRank values with personalization aspects. To achieve personalization, these algorithms need specific input: [8] for example needs a set of personalized hub pages with high PageRank to drive the computation. In this paper we show how to automate this hub selection process and build upon the latter algorithm to implement a platform for personalized ranking. We start from the set of bookmarks collected by a user and extend it to contain a set of hubs with high PageRank *related* to them. To get additional input about the user, we implemented a proxy server which tracks and analyzes user's surfing behavior and outputs a set of pages preferred by the user. This set is then enrichened using our HubFinder algorithm, which finds related pages, and used as extended input for the [8] algorithm. All algorithms are integrated into a prototype of a personalized Web search system, for which we present a first evaluation.

1 Introduction

Using the link structure of the World Wide Web to rank pages in search engines has been investigated heavily in recent years. The success of the Google Search Engine [5, 3] 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 indicates about 4.3 billion pages) lead researchers to explore faster and more personalized page ranking algorithms, in order to address the fact that some topics are covered by only a few thousand pages and some are covered by millions. For many topics, the existing PageRank algorithm is not sufficient to filter the results of a search engine query. Take for example the well-known query with the word “Java” which should return top results for either the programming language or the island in the Pacific: Google definitively prefers the programming language because there are many more important pages on it than on the island. Moreover, most of the existing search engines focus only on answering user queries, although personalization will be more and more important as the amount of information available in the Web increases. Recently, several approaches to solve such problems have been investigated, building upon content analysis or on algorithms which build page ranks personalized for users or classes of users. The most ambitious investigation so far for personalized ranking has been [8], which we have used as starting point for our work.

This paper describes the design and implementation of *PROS*, a personalized ranking platform which uses the algorithm presented in [8] (called the *"PPR algorithm"* – Personalized PageRank – hereafter) plus new algorithms for automated input generation to drive and optimize this algorithm. Our platform is based on HubFinder, an algorithm we developed to find related pages (or hubs, depending on the user) and on a proxy server meant to (temporarily) capture user's surfing behavior. Hubs in this context are Web pages pointing to many other important pages (i.e. with a high rank). Their counterpart are authorities, which are high quality pages pointed by many hubs.

In the original [8] paper, PPR user profiles, used as input for building personalized ranks, are gained by presenting users a set of pages/hubs with high PageRank (as computed using [14]) from which they can choose a set of preferred pages. The disadvantage of this procedure is that this operation takes time and might often be superfluous as most Internet users have some bookmarks of their own already, which could be used to derive their user profile. We therefore wanted to build such a preference set *automatically*, using user's bookmarks and/or most surfed pages (i.e. pages read for a longer period of time, or voted for by a user). This resulting set can then be extended using an algorithm which finds high quality related pages.

The contributions of this paper are: (1) a platform which automates the computation of personalized ranks by generating more comprehensive input data with less user effort, and which consists of two modules: one based on user's bookmarks and the other based on the output of a specialized proxy server which computes the pages most likely to be considered interesting by the user; (2) both modules use HubFinder – a fast and flexible algorithm for finding related pages using the link structure of the World Wide Web, and HubRank – a modified PageRank algorithm which combines the authority value with the hub value of Web pages, in order to further extend these sets of Web pages into the input data needed by the PPR algorithm; and (3) first experimental results from integrating PROS into a personalized Web search system.

We will start by covering the algorithms related to or directly used as background of our research in section 2. Section 3 discusses our PROS platform, which automates the computation of personalized ranks. First experimental results are presented in section 4. Finally, section 5 describes other related Web search systems, and section 6 concludes with discussion of further work.

2 Background

2.1 Web Graph

In the paper we will use a notation similar to [8]. $G = (V, E)$ represents the *Web graph*, where V is the set of all Web pages and E is the set of directed edges $< p, q >$. E contains an edge $< p, q >$ 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 PageRank

PageRank is an algorithm for computing a Web page score based on the graph inferred from the link structure of the Web. The motivating idea for PageRank 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, the PageRank algorithm 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 $\mathbf{PR}$ of page ranks is the eigenvector of A corresponding to its dominant eigenvalue.

Given a Web page p, the PageRank formula is:

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

The dumping factor $c < 1$ (usually 0.15) is necessary to guarantee convergence (A is not irreducible, i.e. G is not strongly connected) and to limit the effect of rank sinks [2]. Intuitively, a random surfer will follow an outgoing link from the current page with probability $(1-c)$ and will get bored and select a different page with probability c.

2.3 Personalized Page Ranking

PageRank. Initial steps towards personalized page ranking are already described by [14] who proposed a slight modification of the PageRank algorithm to redirect the random surfer towards preferred pages using the $\mathbf{E}$ vector. Several distributions for this vector have been proposed since.

Topic-sensitive PageRank. [6] builds a topic-oriented PageRank, starting by computing off-line a set of 16 PageRank vectors based on the 16 main topics of the Open Directory Project [13]. Then, the similarity between a user query and each of these topics is computed, and the 16 vectors are combined using appropriate weights.

Personalized PageRank. *Description*. As the most recent investigation, [8] uses a new approach: it focuses on user profiles. One Personalized PageRank Vector (PPV) is computed for each user. The personalization aspect of this algorithm stems from a *set of hubs* (H), each user having to select her *preferred pages* from it. PPVs can be expressed as a linear combination of basis vectors (PPVs for preference vectors with a single non-zero entry corresponding to each of the pages from P, the preference set), which could be selected from the precomputed basis hub vectors, one for each page from H, the hub set. To avoid the massive storage resources the basis hub vectors would use, they are decomposed into partial vectors (which encode the part unique to each page, computed at run-time) and the hub skeleton (which captures the interrelationships among hub vectors, stored off-line).

Algorithm. In the first part of the paper, the authors present three different algorithms for computing basis vectors: ”Basic Dynamic Programming”, ”Selective Expansion” and ”Repeated Squaring”. In the second part, specializations of these algorithms are combined into a general algorithm for computing PPVs, as depicted below:

1. Compute one partial vector for each page in H using a specialization of the Selective Expansion algorithm.
2. Compute the hubs skeleton using a specialization of the Repeated Squaring algorithm and taking the intermediate results from step 1 as input.
3. Compute the PPV using the computed partial vectors and hubs skeleton, and the preference vector.

For an in-depth view of this process, we refer the reader to [8].

3 PROS: A Personalized Ranking Platform

Introduction. As explained before, personalized rankings can improve current Web search systems by adapting results to user preferences. The algorithm presented in [8] is the most recent step in this direction. An issue left open in [8] is how a set of highly rated hubs, needed as input for the adaptation process, is selected by the user. The personalization (and therefore success) relies on the user’s ability to choose such high quality hubs which match her preferences.

In this section, we describe how to exploit information collected from the user to derive the highly rated hubs that represent the user profile. The computation is performed automatically based on the following *input*:

– *Most surfed pages.* Pages visited by the user are tracked using a specialized proxy we implemented. The proxy records information about the duration the user looked at a page and how frequently she returned to it.

– *User’s bookmarks.* Additionally, we use the user’s bookmarks as an indication for user preferences. Currently, bookmarks are directly provided by the user, but this interaction could also be automated (e.g. using a browser plugin).

Architecture. The PROS platform consists of two main modules, which use the two input sets described above. They use HubFinder and HubRank, two algorithms we developed for finding related pages using the Web link structure and for ranking Web pages, respectively (presented briefly later in this section and in more detail in [4]).

The first module consists of applying the following operations:

1. Get bookmarks from the user.
2. Add bookmarks to the *preference set*.
3. Apply HubFinder, using user’s bookmarks as input and HubRank scores as trimming criterion. HubRank is the best criterion in this situation, because the PPR algorithm needs hubs with high PageRank as input and HubRank has been designed as a biasing of PageRank towards hubs, as discussed later in this section.
4. Add the preference set and the output from the previous step to the *hub set*.

The second module is based on using a proxy server for a limited period of time in order to capture user’s ”surfing behavior”. Its modus operandi is described below:

1. The user surfs the Web using a given proxy. The proxy will output the pages examined by the user for a certain period of time (there must be both a lower threshold and an upper one to avoid the situation when the user leaves the browser open for

a long period of time without using it), as well as those most frequently revisited. The more time it is used, the better ranking accuracy will be aquired.

2. Add the user's most surfed pages (as recorded by the proxy) to the *preference set*.
3. Apply HubFinder with HubRank as criterion and a small radius and number of output pages. We want the pages related to user's bookmarks to be more important than the pages related to his/her most surfed ones and using a smaller radius is a way to achieve this.
4. Add user's most surfed pages, as well as the pages related to them to the *hub set*.

Finally, the PPR algorithm is executed using the newly computed preference and hub sets. The complete process is depicted in figure 1.

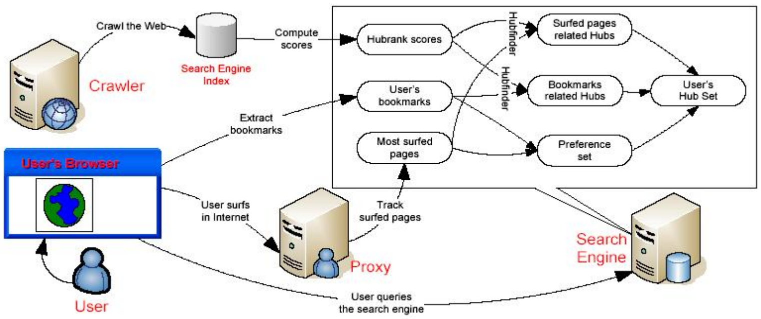


Fig. 1. Personalized Ranking Platform

HubFinder is an algorithm for finding hubs, related to an initial base set of Web pages. We define *related* similarly to [7], 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). We should also add that the distance (the number of links followed) between such two pages is usually less than 6 (according to our experiments, in cases where the distance is bigger the link information becomes insufficient to say that pages are similar in context with a high enough probability), and thus the related hubs are in the vicinity of the starting page. The maximum distance (noted σ and also called radius) is a parameter for HubFinder.

In order to get a good set of related pages we took into account the following aspects: the set has to be small, rich in relevant pages and it should contain many of the strongest authorities. [10] extracts the top results of a query sent to a search engine and builds a focused sub-graph of the WWW around them. It then extends this base set by adding all pages pointed to and at most d pages pointing to each page from it. This operation is called *Kleinberg extension*. The author extends the initial set only once, but he focused on computing Hub and Authority scores, whereas we were focusing on finding related pages or hubs. Therefore we iteratively apply the Kleinberg extension several times on the resulting set of each previous iteration in order to obtain more pages and thus more representative results. As this scenario leads to very big output sets (up to

500,000 pages), trimming is necessary after each intermediate step. The pseudo-code of the resulted HubFinder algorithm is as follows:

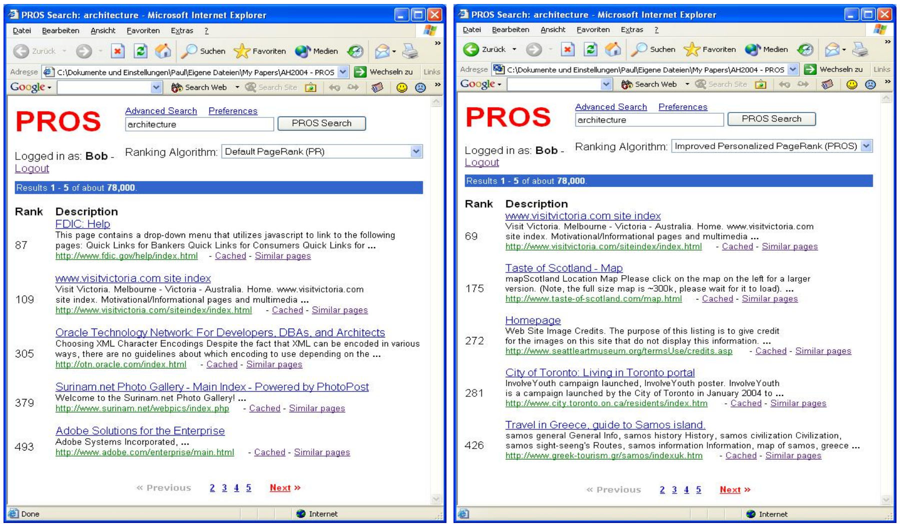
```
Let $\Gamma$ be the Base Set of pages whose related hubs we are looking for
$\Gamma \leftarrow$ Apply the Kleinberg Extension on $\Gamma$ once
$\Gamma' \leftarrow \Gamma$
For $i = 1$ to $\sigma$ do:
    $\Gamma'' \leftarrow$ Apply the Kleinberg Extension on $\Gamma'$ once
    Trim $\Gamma''$ to contain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

Two aspects have to be considered: how many pages should we keep after each iteration and which are the *interesting pages*? Regarding the former one, we keep one percent of the current set size, whereas the best criterion tackling the latter issue are global rank scores (e.g. as computed with PageRank or a similar algorithm). [4] contains an in-depth discussion of the algorithm, a formula on the exact number of pages to keep, as well as a proposed extension based on text analysis.

HubRank. We started from the idea that a page pointing to a good hub is a candidate for having a high hub rank as well. Often 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 compute these hub scores, we modified the PageRank personalization vector to consider the out-degree of the pages. Intuitively, the random surfer prefers 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. In PageRank, the vector **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 **E** to $E_i = |O(i)|\frac{NP}{|O|}$ where $|O|$ is the summation of the out-going links over the whole Web graph. As PageRank focuses on authorities in the first part of the formula, we focused on hubs in the second part.

Prototype. Current Web search systems apply only basic personalization techniques (e.g. presenting a user interface in Spanish if the access is from a Spanish IP address). However, this refers only to how the search engine interacts with the user, but it uses the same ranking process no matter who submits the query. To exemplify this problem, let us imagine that a user searches using the keyword ”architecture”. Output topics may vary from computer architecture to building architecture or even something else. By extracting user’s interests from her bookmarks (if she likes building architecture she would have some bookmarks on it) and from her most visited pages (she would check buiding architecture pages often), we can create a personalized view of the global ranks, and thus provide tailored output for each user. A screenshot of our prototype can be seen in figure 2. As a comparison, we present the results obtained when ranking URLs with the PageRank algorithm [14] on the left side, and with PROS on the right. Our tester was interested in building architecture. While with PageRank only two output URLs were relevant, all five generated by PROS were worth checking.


PROS Search: architecture - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück
Suchen
Favoriten
Medien
Adresse
C:\Dokumente und Einstellungen\Paul\Eigene Dateien\My Papers\AH2004 - PROS
Wechseln zu
Links
Google
Search Web
Search Site
PROS
Advanced Search
Preferences
PROS Search
Logout
« Previous
2 3 4 5
Next »
Done
Internet
PROS Search: architecture - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück
Suchen
Favoriten
Medien
Adresse
C:\Dokumente und Einstellungen\Paul\Eigene Dateien\My Papers\AH2004 - PROS
Wechseln zu
Links
Google
Search Web
Search Site
PROS
Advanced Search
Preferences
PROS Search
Logout
« Previous
2 3 4 5
Next »
Internet

Search: architecture

Logged in as: **Bob** -

Ranking Algorithm: Default PageRank (PR)

Results **1** - **5** of about **78,000**.

Rank	Description
87	FDIC: Help This page contains a drop-down menu that utilizes javascript to link to the following pages: Quick Links for Bankers Quick Links for Consumers Quick Links for **...** http://www.fdic.gov/help/index.html - Cached - Similar pages
109	www.visitvictoria.com site index Visit Victoria. Melbourne - Victoria - Australia. Home. www.visitvictoria.com site index. Motivational/Informational pages and multimedia **...** http://www.visitvictoria.com/siteindex/index.html - Cached - Similar pages
305	Oracle Technology Network: For Developers, DBAs, and Architects Choosing XML Character Encodings Despite the fact that XML can be encoded in various ways, there are no guidelines about which encoding to use depending on the **...** http://otn.oracle.com/index.html - Cached - Similar pages
379	Surinam.net Photo Gallery - Main Index - Powered by PhotoPost Welcome to the Surinam.net Photo Gallery! **...** http://www.surinam.net/webpics/index.php - Cached - Similar pages
493	Adobe Solutions for the Enterprise Adobe Systems Incorporated, **...** http://www.adobe.com/enterprise/main.html - Cached - Similar pages

Search: architecture

Logged in as: **Bob** -

Ranking Algorithm: Improved Personalized PageRank (PROS)

Results **1** - **5** of about **78,000**.

Rank	Description
69	www.visitvictoria.com site index Visit Victoria. Melbourne - Victoria - Australia. Home. www.visitvictoria.com site index. Motivational/Informational pages and multimedia **...** http://www.visitvictoria.com/siteindex/index.html - Cached - Similar pages
175	Taste of Scotland - Map mapScotland Location Map Please click on the map on the left for a larger version. (Note, the full size map is ~300k, please wait for it to load). **...** http://www.taste-of-scotland.com/map.html - Cached - Similar pages
272	Homepage Web Site Image Credits. The purpose of this listing is to give credit for the images on this site that do not display this information. **...** http://www.seattleartmuseum.org/termsUse/credits.asp - Cached - Similar pages
281	City of Toronto: Living in Toronto portal InvolveYouth campaign launched, InvolveYouth poster. InvolveYouth is a campaign launched by the City of Toronto in January 2004 to **...** http://www.city.toronto.on.ca/residents/index.htm - Cached - Similar pages
426	Travel in Greece, guide to Samos island. samos general General Info, samos history History, samos civilization Civilization, samos sight-seeng's Routes, samos information Information, map of samos, greece **...** http://www.greek-tourism.gr/samos/indexuk.htm - Cached - Similar pages

Fig. 2. Prototype of the PROS Web Search System

4 Experiments and Results

Input Data. We performed tests on several small Web crawls (3 to 40 thousand pages) and on two bigger ones, one with one million and one with three million Web pages. The results presented in this paper use the largest set. Furthermore, we ran PPR and PROS using several data sets as input and several users, but selected only the most representative experiments here because of space constraints.

Algorithm	Preference Set	Hub Set
PPR	30 user defined bookmarks.	User's bookmarks (30) plus top ranked PageRank pages. Totally about 1200 pages.
PROS	30 user defined bookmarks plus 78 pages selected tracking user's surfing behavior (108 pages in total).	The preference set plus its related pages plus top ranked PageRank pages. Totally about 1700 pages.

Table 1. Input Data for the PPR algorithm experiments

Our first experiment follows all guidelines of the original paper. It has 30 user bookmarks as preference set and a hub set mixing user's bookmarks with top PageRank documents. The second experiment uses the input we obtained with our ranking platform. A tester surfed the Web for about two weeks using our proxy and we selected 78 pages as her ”fingerprint”. These were merged with her 30 bookmarks (same as in the first experiment) into the *preference set*. Then, we applied HubFinder with HubRank as criterion on both the set of bookmarks and the set of most surfed pages, obtaining about 900 pages from the former one and about 400 from the latter one (we used a radius of 5 for the bookmarks and a radius of 2 for the most surfed pages). To these 1300 pages some top PageRank pages were added and the resulting set was used as *hub set*. A description of the input data used can be found in table 1. Our tester was an architect, having traveling and software as other hobbies, and sports as a secondary interest. Her bookmarks were distributed accordingly: 15 on architecture, 7 on traveling, 6 on software and 2 on sports.

Results. To analyze the resulting ranks, we selected some general keywords (see table 2) and performed Web searches, exactly as in a search engine. Results were sorted with respect to their ranks, without considering term frequency of keywords in output documents. The ranking algorithms used were PageRank, PPR, and PROS. Although the first one does not involve any personalization, we decided to implement it too, as it is the most popular algorithm and useful as background for our evaluation.

Query Keywords	**PageRank**			**PPR**			**PROS**		
	Rel.	Par. Rel.	Irrel.	Rel.	Par. Rel.	Irrel.	Rel.	Par. Rel.	Irrel.
architecture	5	3	2	3	7	0	8	2	0
building	3	2	5	2	3	5	4	1	5
Paris	6	0	4	2	3	5	6	2	2
park	6	0	4	8	0	2	10	0	0
surf	3	0	7	4	2	4	7	2	1
Total	23	5	22	19	15	16	35	7	8

Table 2. Relevancy value for different search keywords and different algorithms

The top 10 URLs obtained by ranking the search results with each algorithm were classified into the following categories: (a) Relevant (denoted by ”Rel.” in table 2) if the URL was on one of the four topics of interest of our tester; (b) Partially Relevant (noted as ”Par.Rel.”) if it was on a topic related to one of the above-mentioned four ones (e.g. an URL on hardware architectures was considered partially related to computer software); or (c) Irrelevant (”Irrel.” in table 2) if it was not in any of the previous categories. A detailed list with all the output URLs can be found in [15].

Discussion. The most important issue is that, as expected, the original PageRank algorithm provides top results on several topics, even though the searcher is almost always interested in only a specific one. This behavior is understandable, as the algorithm cannot disambiguate results based on user preferences.

The PPR algorithm performs only slightly better in this experiment (the total number of possibly relevant URLs is 34, whereas for PageRank it is 28), mostly because the input sets were too sparse and qualitatively not very good. This might be improved by adding additional top PageRank pages to the preference set, but we did not use this approach, as it would have definitely damaged the personalization aspect (remember that top PageRank pages can be on any topic).

Finally, we see significant improvements when using PROS. The number of relevant pages is much higher than for the other two algorithms. However, we still got some bad output URLs (e.g. for the search keyword ”building”). We think this happened because of the general profile of our tester. Other tests performed with more focused testers (i.e. with a profile on exclusively one topic, such as ”architecture”) provided even better results, but we consider the experiment presented in this paper to be the closest to a real-life situation.

As we know from [8], pages from the preference set may have different importance. In our tests, all pages had the same weight, that is $\frac{1}{|PS|}$, where PS is the preference set. Let us denote by B the set of bookmarks and S the set of user's most surfed pages. In this case, we can give for example $\frac{2}{3*|B|}$ importance to bookmarks and $\frac{1}{3*|S|}$ to user's most surfed pages. We think this approach (or a more complex one which automatically gives an importance value to each of the most surfed pages, depending on the amount of surfing time or revisits) would provide even more accurate ranks. Further experiments are needed to define the correct biasing of these pages.

Generally, our experiments allow us to conclude that PROS increases the ranks of pages similar to user's bookmarks, as well as those that are most likely to be considered interesting by the user (and thus the granularity of relevant results when performing a Web search). If the tester uses the proxy server for longer periods, the accuracy of the latter set is proportionately bigger (i.e. the size of the ”most surfed pages” set is bigger, and therefore the rankings are more accurate).

5 Other Web Search Systems

Research on Web searching usually focused on algorithms, as they comprise the most important part of such an engine. However, these algorithms are not always straightforward to integrate into a search engine and attention must also be paid to the user - search engine interaction. [12] for example aims at satisfying all types of users (either experienced, or novice) by translating the search goal into good query keywords. Although search goals must be manually disambigued (e.g. solve a problem, find people who, etc.), the system allows users to type queries in natural language and produces the search keywords automatically using an inference engine.

[11] is somehow similar to PROS in the sense that user profiles are built automatically and Web pages are recommended to users based on personalization approaches. However, its contributions are orthogonal to ours: surfers are guided towards relevant pages while they are browsing judging on the pages they have previously visited, whereas we focus on integrating such personalization into page ranks used by a search engine. [16] presents a new system for computing Web page ranks in a distributed fashion, which could be nicely extended by our PROS approach.

Another interesting approach is described in [1], which adds personalization to the JobFinder [9] search engine. It uses profile jobs and Information Retrieval operators to classify retrieved jobs as relevant or irrelevant.

6 Conclusions and Further Work

We presented the design and implementation of a personal ranking platform (PROS), providing personalized ranking of Web pages based on user preferencees, while automating the input generation process for the PPR algorithm [8]. To achieve this, we use user bookmarks as well as a record about the user's surfing behavior (gained with a proxy), which are then extended using our new HubFinder algorithm with HubRank scores as trimming criterion. HubRank score is based on PageRank but additionally takes hub quality of pages into account.

We are currently extending our tests towards more users and different environments in order to get in depth evaluations of our platform. Furthermore, we are investigating the additional inclusion of collaborative filtering algorithms to PROS, which would allow us to also use personalize ranking based on groups of similar users instead of single users, to further increase the input set available for the personalization algorithm.

References

1. K. Bradley, R. Rafter, and B. Smyth. Case-based user profiling for content personalization. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 133–143. Springer-Verlag, 2000.
2. Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.
3. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
4. Paul-Alexandru Chirita, Daniel Olmedilla, and Wolfgang Nejdl. Pros: A personalized ranking platform for web search. Technical report, L3S and University of Hannover, Feb 2004.
5. Google search engine. http://www.google.com.
6. T. Haveliwala. Topic-sensitive pagerank. In *In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii*, May 2002.
7. G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, 2002.
8. G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
9. Jobfinder search engine. http://www.jobfinder.com.
10. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
11. Nicholas Kushmerick, James McKee, and Fergus Toolan. Towards zero-input personalization: Referrer-based page prediction. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2000.
12. Hugo Liu, Henry Lieberman, and Ted Selker. Goose: A goal-oriented search engine with commonsense. In *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 253–263. Springer-Verlag, 2002.
13. Open directory project. http://dmoz.org/.
14. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.
15. Pros project home page. http://www.learninglab.de/˜chirita/pros/pros.html.
16. Jie Wu. Towards a decentralized search architecture for the web and p2p systems. In *Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems Workshop held at the HyperText03 Conference*, 2003.