Finding Hubs for Personalized Web Search
Different ranks to different users

Daniel Olmedilla
Overview

- Introduction
- State of the Art
- Our Work
- Results
- Conclusions
Overview

- Introduction
- State of the Art
- Our Work
- Results
- Conclusions
Why is it interesting?

- Existing ranking algorithms do not take into account user knowledge, preferences or interests.

- One ranking algorithm for the whole set of people using a web search engine does not seem to be enough.

- At the moment, there is not any web search engine offering real personalized web search for any user.
Current process

- User submits a query
- The query is processed
- Search into the index for relevant documents
- Rank of the results
- Show the results
Improvements of the process (I)

- Semantic Web approaches
  - Improvement of document “relevance”
  - Insert embedded metadata into html page
    - Problem: it is very difficult to change 1 billion web pages and there are many inexpert users
  - Semi-automatic annotation for the entire WWW
    - Problem: difficult to classify the whole content of WWW. There is no standard yet
  - It needs to share the user profile in runtime during the search. Privacy problems.
Improvements of the process (II)

- Our approach
  - Improvement of ranking
  - No need for changes into web pages
  - Solution between current search engines and Semantic Web approaches
  - Each user can have his own and more effective ranking
    - Based on his preferences and interests
  - Reduction of the time needed by the user for search
Overview

- Introduction

- State of the art
  - Crawlers and Search Engines
  - Ranking algorithms
  - Personalization

- Our work

- Results

- Conclusions
The whole process

- Crawling
  - Retrieval of information from the web

- Indexing
  - Storage of information

- Searching
  - Search into the stored information

- Ranking
  - Rank the results according to “importance”
Typical search engine (I)
Typical search engine (II)

Elements:
- User Interface: needed to take the user query
- Index: database/repository with the data to be searched
- Search module:
  - Transform query to understandable format
  - Do matching with the index
  - Return the results as output with information needed
- Evaluation module: take care of the user behaviour with the results of the query
Typical crawler structure (I)
Typical crawler structure (& II)

Elements:
- Retrieving module: retrieve each document from the web and give it to the Process module
- URL Listing module: has a list (ordered or unordered) of URLs and give them to the Retrieving module
- Process module: realize the next steps and after that, it gives the result (data) to the Format & Store module:
  - Automatic text analysis
  - Classification
  - Filter
- Format & Store module: convert data to better format and store it into the index
- Index: database/repository with the useful data retrieved
Ranking Algorithms

- PageRank
- HITS
- SALSA
- Randomized HITS
- SimRank
PageRank

- Successfully used in Google
- Importance: page $i$ pointing to page $j$ means vote from $i$ to $j$

$$PR(p) = (1-c) \sum_{q \in O_p} \frac{PR(q)}{O_q} + cE(p)$$
- Concept of Random Surfer
PageRank Example

CO: Company
PA: Project A
PB: Project B
PC: Project C
PL: Project List
RA: Researcher A
RB: Researcher B
RC: Researcher C
UA: University A
UB: University B
UL: University List
HITS (I)

- Hypertext Induced Topic Selection

- Two scores:
  - *Authorities*: pages that have “authoritative” information about topics
  - *Hubs*: pages that have links to many important pages for the same topic

- Solve the problem of “airline company” query

- Three steps:
  - Select starting set of pages
  - Extend starting set of pages
  - Calculate the scores
HITS (II)

Subgraph($\sigma, \xi, t, d$)

$\sigma$: a query string.
$\xi$: a text-based search engine
$t, d$: natural numbers.
Let $R_\sigma$ denote the top $t$ results of $\xi$ on $\sigma$.

Set $S_\sigma := R_\sigma$
For each page $p \in R_\sigma$

Let $\Gamma^+(p)$ denote the set of all pages $p$ points to.
Let $\Gamma^-(p)$ denote the set of all pages pointing to $p$.
Add all pages in $\Gamma^+(p)$ to $S_\sigma$
if $|\Gamma^-(p)| \leq d$, then
Add all pages in $\Gamma^-(p)$ to $S_\sigma$
Else
Add an arbitrary set of $d$ pages from $\Gamma^-(p)$ to $S_\sigma$
End
Return $S_\sigma$

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

$$h^p \leftarrow \sum_{q: (p,q) \in E} a^q$$
HITS Example
Personalized PageRank

\[ PR(p) = (1 - c) \sum_{q \in O_p} \frac{PR(q)}{O_q} + cE(p) \]

- Vector E is the basis of the personalization in PageRank
- It is not possible to have a different vector for each user
  - Storage limit
  - Time limit
Topic-sensitive PageRank

- 16 top-level categories from Open Directory Project
- 16 biased PageRank vectors
- Query-sensitive importance score

\[ P(c_i \mid q) = \frac{P(c_i) \cdot P(q \mid c_i)}{P(q)} \alpha P(c_i) \cdot \prod_j P(q_j \mid c_i) \]

\[ s_{qd} = \sum_i P(c_i \mid q) \cdot r_{id} \]
Scaling Personalized Web Search

- Challenges in Personalized PageRank
  - Reduce storage required
  - Reduce time for computation

- Personalized PageRank Vector (PPV)

- Split PPVs in common part and user specific part
  - Common part is precomputed and stored once
Overview

- Introduction
- State of the art
- Our work
  - WebBase
  - Ranking Algorithms
  - HubRank
  - Proxy
  - HubFinder
  - Personalized Ranks
- Results
- Conclusions
WebBase enhancements

- Link structure generation
  - 2 structures
    - Urls
    - Links

- Restart over an existing link structure
  - Incremental sets of pages crawled

- Performance optimisation
  - Remove of process
  - Remove of storage
Ranking algorithms implementation

Basic to understand how they work and for comparison purposes

Implemented algorithms:
- PageRank
- HITS
- SALSA
- Randomized HITS
- SimRank
HubRank

- Bringing HITS ideas to PageRank

\[ PR(p) = (1 - c) \sum_{q \in O_p} \frac{PR(q)}{O_q} + cE(p) \]

- Using \( E(p) = O_p \frac{n}{|O|} \)
Proxy

- User’s interests are determined by
  - Bookmarks
  - Most surfed pages

- Proxy tracks user behaviour
  - Pages visited
  - Time user was looking at each page
  - Other information can be further develop

- Current assumption
  - One user for IP address
HubFinder (I)

- Based on Kleinberg extension algorithm:
- Filtering at each iteration
- Improvements over the Kleinberg extension algorithm:
  - Better performance
  - Better scalability
HubFinder (II)

Subgraph(Γ, k)

Γ: initial set of pages
k: number of iterations
Let KE be a function that applies the Kleinberg Extension once
Let Sel1 and Sel2 be functions that select important pages

Γ = KE(Γ)
Γ̃ = Γ
For i = 1 to k

Γ* = KE(Γ̃)
Γ* = Sel1(Γ*)
Γ = Γ + Γ*
Γ̃ = Γ*
Γ = Sel2(Γ)
return Γ
Personalized Ranks

- Two problems
  - Given set of hubs to choose
  - Level of personalization depends on the size of the set

- User provides the representative set of pages

- We provide a personalized set obtained from the user input

- User does not have to choose among given hubs (even he does not need to know what is a hub)
Summary of the process

1. Crawl
2. Calculate Ranks
3. Get bookmarks from user
4. Get user’s surfing information (Proxy)
5. Find the important hubs (HubFinder)
6. Construction of preference set & the hub set
7. Apply personalized rank
Overview

- Introduction
- State of the art
- Our work
- Results
  - WebBase
  - HubRank
  - HubFinder
  - Personalized Ranks
- Conclusions
WebBase

- Three executions of the crawler
  - 4,106 different domains
  - 729,384 pages
  - 3,587,842 links

- New crawler execution
  - Another 4,887 domains extracted from previous executions
  - Already crawled around 800,000 new pages
### HubRank

<table>
<thead>
<tr>
<th>Node</th>
<th>PageRank</th>
<th>HITS</th>
<th>HubRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>0.15</td>
<td>0.09</td>
<td>0.15</td>
</tr>
<tr>
<td>PA</td>
<td>0.65</td>
<td>0.31</td>
<td>1.13</td>
</tr>
<tr>
<td>PB</td>
<td>3.13</td>
<td>0.00</td>
<td>2.83</td>
</tr>
<tr>
<td>PC</td>
<td>0.37</td>
<td>0.00</td>
<td>0.30</td>
</tr>
<tr>
<td>PL</td>
<td>0.33</td>
<td>0.58</td>
<td>0.73</td>
</tr>
<tr>
<td>RA</td>
<td>0.42</td>
<td>0.25</td>
<td>0.73</td>
</tr>
<tr>
<td>RB</td>
<td>2.81</td>
<td>0.27</td>
<td>2.28</td>
</tr>
<tr>
<td>RC</td>
<td>0.15</td>
<td>0.49</td>
<td>0.31</td>
</tr>
<tr>
<td>UA</td>
<td>0.37</td>
<td>0.27</td>
<td>0.67</td>
</tr>
<tr>
<td>UB</td>
<td>0.37</td>
<td>0.33</td>
<td>0.67</td>
</tr>
<tr>
<td>UL</td>
<td>0.15</td>
<td>0.11</td>
<td>0.31</td>
</tr>
</tbody>
</table>

![Graph](image)

**Legend:**
- CO: Company
- PA: Project A
- PB: Project B
- PC: Project C
- PL: Project List
- RA: Researcher A
- RB: Researcher B
- RC: Researcher C
- UA: University A
- UB: University B
- UL: University List
HubFinder

- Ranking algorithms comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>PageRank</th>
<th>HITS</th>
<th>SALSA</th>
<th>Randomized HITS</th>
<th>HubRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>632</td>
<td>334</td>
<td>334</td>
<td>676</td>
<td>650</td>
</tr>
</tbody>
</table>

- Better performance than Kleinberg extension Algorithm

- Randomized HITS and HubFinder are the best ones
## Personalized Ranks

<table>
<thead>
<tr>
<th>URL</th>
<th>Original</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.trendmicro.com/en/home/global/legal.htm">www.trendmicro.com/en/home/global/legal.htm</a></td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td><a href="http://www.trendmicro.com/en/about/privacy/overview.htm">www.trendmicro.com/en/about/privacy/overview.htm</a></td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td><a href="http://www.amoebo.com">www.amoebo.com</a></td>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td><a href="http://www.amoebo.com/index.asp">www.amoebo.com/index.asp</a></td>
<td>4</td>
<td>*</td>
</tr>
</tbody>
</table>

...  

<table>
<thead>
<tr>
<th>URL</th>
<th>Original</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.domus3d.com/default.asp">www.domus3d.com/default.asp</a></td>
<td></td>
<td>66</td>
</tr>
<tr>
<td><a href="http://www.gardenvisit.com">www.gardenvisit.com</a></td>
<td>*</td>
<td>9</td>
</tr>
<tr>
<td><a href="http://www.urban-advantage.com/index.html">www.urban-advantage.com/index.html</a></td>
<td>*</td>
<td>24</td>
</tr>
<tr>
<td><a href="http://www.metropolismag.com/html/designmart/index.html">www.metropolismag.com/html/designmart/index.html</a></td>
<td>*</td>
<td>29</td>
</tr>
<tr>
<td>phillipsplanning.com/pastprojects1.html</td>
<td>*</td>
<td>72</td>
</tr>
<tr>
<td><a href="http://www.museumofarchitecture.com/main.html">www.museumofarchitecture.com/main.html</a></td>
<td>*</td>
<td>3</td>
</tr>
<tr>
<td>deck-porch-gazebo-plan.vdeck.com</td>
<td>*</td>
<td>30</td>
</tr>
<tr>
<td><a href="http://www.wrdesign.com/gallery.html">www.wrdesign.com/gallery.html</a></td>
<td>*</td>
<td>21</td>
</tr>
<tr>
<td><a href="http://www.architekturmuseum.ch/htmls/indexe.htm">www.architekturmuseum.ch/htmls/indexe.htm</a></td>
<td>*</td>
<td>70</td>
</tr>
<tr>
<td><a href="http://www.bauhaus.de/english/museum/index.htm">www.bauhaus.de/english/museum/index.htm</a></td>
<td>25790</td>
<td>69</td>
</tr>
<tr>
<td><a href="http://www.bauhaus.de/english/bauhaus1919/arkitektur/index.htm">www.bauhaus.de/english/bauhaus1919/arkitektur/index.htm</a></td>
<td>26457</td>
<td>287</td>
</tr>
<tr>
<td>chi-athenaeum.org/gdesign/gdesin0.htm</td>
<td>*</td>
<td>50</td>
</tr>
<tr>
<td><a href="http://www.archiradar.com/eng/index.htm">www.archiradar.com/eng/index.htm</a></td>
<td>*</td>
<td>44</td>
</tr>
<tr>
<td>gdalliance.com</td>
<td>*</td>
<td>478</td>
</tr>
<tr>
<td><a href="http://www.astechsoft.com/gpsandreiw.htm">www.astechsoft.com/gpsandreiw.htm</a></td>
<td>*</td>
<td>86</td>
</tr>
<tr>
<td><a href="http://www.fatcad.com/home.asp">www.fatcad.com/home.asp</a></td>
<td>*</td>
<td>11</td>
</tr>
<tr>
<td><a href="http://www.command-digital.com/panoramal.htm">www.command-digital.com/panoramal.htm</a></td>
<td>*</td>
<td>19</td>
</tr>
<tr>
<td><a href="http://www.contractcaddgroup.com/3d.htm">www.contractcaddgroup.com/3d.htm</a></td>
<td>*</td>
<td>79</td>
</tr>
<tr>
<td><a href="http://www.e-magine.ws/products.htm">www.e-magine.ws/products.htm</a></td>
<td>*</td>
<td>100</td>
</tr>
<tr>
<td><a href="http://www.atlanticarchitects.com/about.htm">www.atlanticarchitects.com/about.htm</a></td>
<td>*</td>
<td>115</td>
</tr>
<tr>
<td><a href="http://www.architecture.com/go/Architecture/LReference/Library_899.html">www.architecture.com/go/Architecture/LReference/Library_899.html</a></td>
<td>26954</td>
<td>34</td>
</tr>
</tbody>
</table>

...  

<table>
<thead>
<tr>
<th>URL</th>
<th>Original</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.sun.com/downloads</td>
<td>4405</td>
<td>171</td>
</tr>
</tbody>
</table>
Overview

- Introduction
- State of the art
- Our work
- Results
- Conclusions
Conclusions & further work

- Conclusions
  - Better rank algorithm based on properties of PageRank and HITS
  - Fast and focused algorithm to find related hubs
  - Personalization from user input and behaviour

- Further work
  - Increase the size of the crawl
  - SimRank improvement
  - More information from the user
  - More experiments and evaluation
Publications

- Accepted as a poster at 1st Latin American Web Congress (Santiago, Nov. 2003)
  - http://www.la-web.org/acceptedPosterTrack.html
- Comments from reviewers
  - Originality: Weak Accept/Accept
  - Significance: Accept/Accept
  - Technical quality: Neutral/Accept
  - Relevance: Accept/Accept
- I believe that the proposed algorithm, HUBFINDER (Pagerank+HITS) should be tested on larger databases. In this work only 10,000 pages were used.
- A new algorithm is presented. It has properties of pagerank and HITS. It is faster than the others.
- The paper is clear and explicit. They compare the algorithm with HITS.