CIS 833 – Information Retrieval and Text Mining

Lecture 18

Link Analysis

November 3, 2015

Credits for slides: Allan, Arms, Manning, Lund, Noble, Page.

Next

- Web Search
  - Textbook Chapter 21 – Web analysis
  - Monika R. Henzinger, Hyperlink Analysis for the Web. IEEE Internet Computing, vol. 5, no. 1, pp. 45-50, Jan/Feb., 2001.

Connectivity-Based Ranking

Ranking based on hyperlink analysis

- Query-independent ranking
  - PageRank: authorities
- Query-dependent ranking
  - HITS: authorities and hubs

- *Authorities* are pages that are recognized as providing significant, trustworthy, and useful information on a topic.
- *Hubs* are index pages that provide lots of useful links to relevant content pages (topic authorities).

PageRank Algorithm

Let S be the total set of pages and n=|S|, i.e. n is the number of Web pages in the collection

Choose ε s.t. $0<\varepsilon<1$, e.g. 0.15

Initialize $\forall A \in S$: $R(A) = 1/n$

Until ranks do not change (much) (*convergence*)

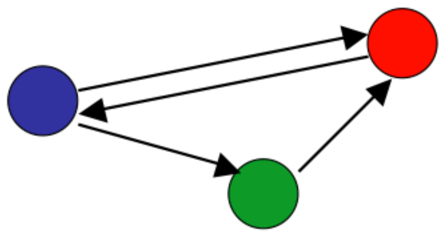
For each $A \in S$:

$$R'(A) = \left[(1-\varepsilon) \sum_{B \to A} \frac{R(B)}{out(B)} \right] + \frac{\varepsilon}{n}$$

$$c = 1 / \sum_{A \in S} R'(A)$$

For each $A \in S$: $R(A) = cR'(A)$ (*normalize*)

PageRank Exercise



MapReduce Implementation

//Y is a page, PR(Y) is current PageRank of Y, and $Z_1,\ldots,Z_n$ are outgoing links from Y

Map input: $\left(Y,[PR(Y),\{Z_1,\ldots,Z_n\}]\right)$

Map output: $\left(Z_i,\dfrac{PR(Y)}{n}\right),\left(Y,\{Z_1,\ldots,Z_n\}\right)$

Reduce input: $\left(Y,[S_1,\ldots,S_m,\{Z_1,\ldots,Z_n\}]\right)$

Reduce output: $\left(Y,\left[\dfrac{\varepsilon}{N}+(1-\varepsilon)\sum_{i=1}^{m}S_m,\{Z_1,\ldots,Z_n\}\right]\right)$

MapReduce Implementation

```
public void map(LongWritable key, Text value,Context context)
{
  …
   extract page from value
   extract links from value
  …

  context.write(new Text(page), new Text(links));

  tokenize links

  while(tokenizer.hasMoreElements())
  {
     String outLink = tokenizer.nextToken().toString().trim();
     context.write(new Text(outLink), new Text(Double.toString((double)rank/(double)(totalOutLinks))));
   }
}
```

PageRank Issues

- How realistic is the random surfer model?
  - What if we modeled the back button? [Fagi00]
  - Surfer behavior sharply skewed towards short paths [Hube98]
  - Search engines, bookmarks & directories make jumps non-random.

PageRank Retrieval

- Preprocessing:
  - Given graph of links, compute the rank of each page A.
- Query processing:
  - Retrieve pages meeting query.
  - Rank them by their PageRank.
  - Order is query-*independent*.

PageRank-Biased Spidering

- Use PageRank to direct (focus) a spider on “important” pages.
- Compute page-rank using the current set of crawled pages.
- Order the spider's search queue based on current estimated PageRank.

Topic Specific PageRank [Haveliwala 02]

- Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:
  - Selects a category (say, one of the 16 top level ODP categories) based on a query
  - Teleport to a page uniformly at random within the chosen category
- Sounds hard to implement: can't compute PageRank at query time!

Topic Specific PageRank [Haveliwala 02]

Implementation

- **offline**: Compute PageRank distributions wrt *individual* categories
  - Query independent model as before
  - Each page has multiple PageRank scores – one for each ODP category, with teleportation only to that category
- **online**: Distribution of weights over categories computed by query context classification
  - Calculate the similarity of the query to each of the ODP categories
  - Generate a dynamic PageRank score for each page - weighted sum of category-specific PageRanks

"Personalized" PageRank

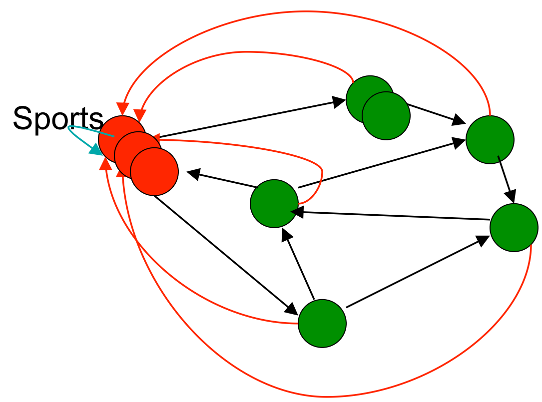
- PageRank can be biased (personalized) by changing the teleporting distribution P(A)=1/n to a non-uniform distribution.
- Restrict "random jumps" to a set of specified relevant pages.
- For example, let $P(A) \sim 0$ except for one's own home page, for which $P(A) = \alpha$.
- This results in a bias towards pages that are closer in the web graph to your own homepage.

Influencing/Personalizing PageRank

- Input:
  - Web graph *W*
  - Influence vector **v**

    v : (topic → degree of influence)
- Output:
  - Rank vector **r**: (page → page importance wrt **v**)
  - **r** = PR(*W* , **v**)
- Assumption: interests can be approximated as a combination of a small number of topic page distributions.

Non-uniform Teleportation



Teleport with 10% probability to a Sports page

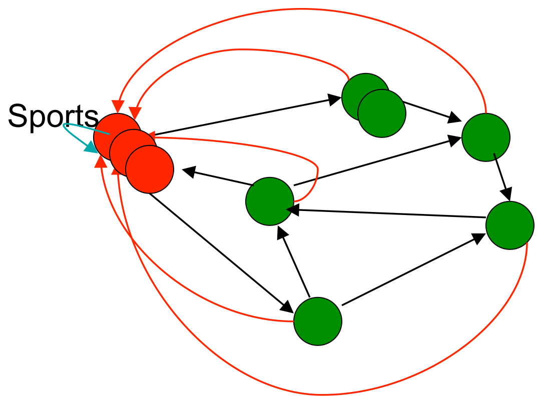
Composite Score

- For a set of personalization vectors $\{\mathbf{v}_j\}$

$$\sum_j [w_j \cdot PR(W, \mathbf{v}_j)] = PR(W, \sum_j [w_j \cdot \mathbf{v}_j])$$

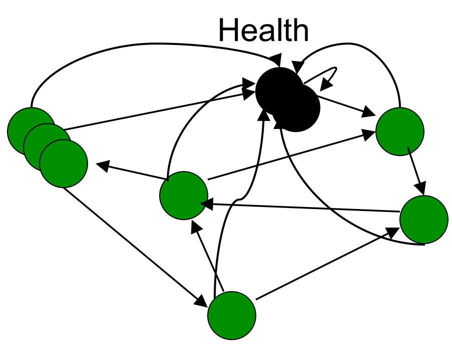
- Weighted sum of rank vectors itself forms a valid rank vector, because PR() is linear wrt $\mathbf{v}_j$

Interpretation



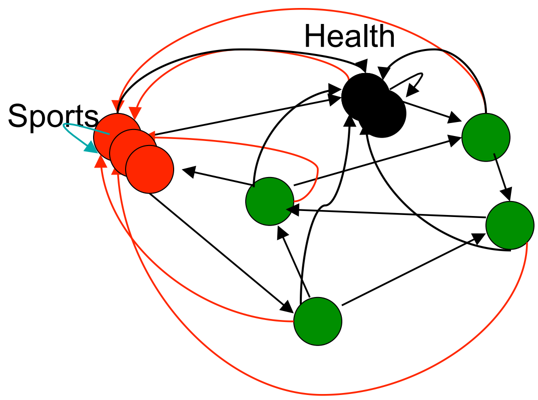
10% Sports teleportation

Interpretation



10% Health teleportation

Interpretation



PR = (0.9 PR_{sports} + 0.1 PR_{health}) gives you:
9% sports teleportation, 1% health teleportation

PageRank Conclusions

- Link analysis uses information about the structure of the web graph to aid search.
- It is one of the major innovations in web search.
- It is the primary reason for Google's success.

- According to them…
  - "The heart of our software is PageRank™, a system for ranking web pages developed by our founders…. And while we have dozens of engineers working to improve every aspect of Google on a daily basis, PageRank continues to play a central role in many of our web search tools." [http://www.google.com/technology]

Google Ranking

- PageRank is used in Google, but so are many other clever heuristics.
- Complete Google ranking includes (based on university publications prior to commercialization).
  - Vector-space similarity component.
  - Keyword proximity component.
  - HTML-tag weight component (e.g., title preference).
  - PageRank component.
- Details of current commercial ranking functions are trade secrets.

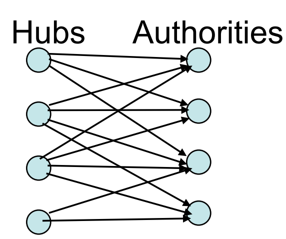
HITS algorithm

Hyperlink-Induced Topic Search (HITS)

- Algorithm developed by Kleinberg in 1998, as part of IBM's Clever search project.
- Attempts to computationally determine hubs and authorities on a particular topic through analysis of a relevant subgraph of the web.
- Based on mutually recursive facts:
  - Hubs point to lots of authorities.
  - Authorities are pointed to by lots of hubs.
- In response to a query, instead of an ordered list of pages each meeting the query, find <u>two</u> sets of inter-related pages: *hub pages* and *authority pages.*
- Best suited for "broad topic" queries rather than for page-finding queries.
- Gets at a broader slice of common *opinion*.

Hubs and Authorities

- Together they tend to form a bipartite graph:

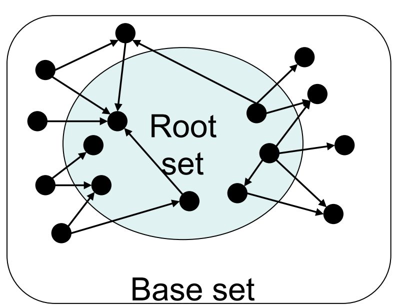


HITS Algorithm

- Computes hubs and authorities for a particular topic specified by a query.
- First determines a set of relevant pages for the query called the *base* set *S*.
- Analyzes the link structure of the web subgraph defined by *S* to find authority and hub pages in this set.

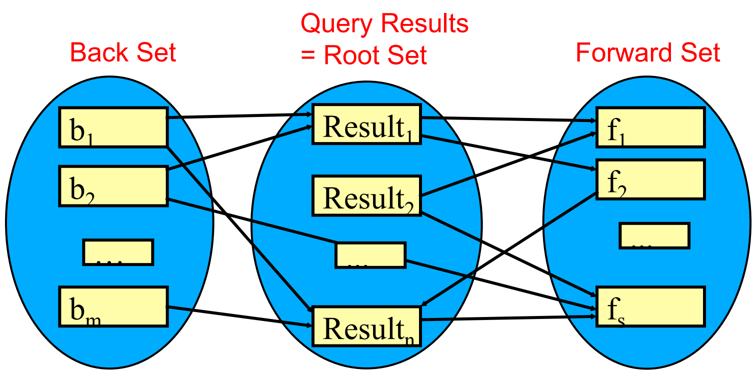
Constructing a Base Subgraph

- For a specific query *Q*, let the set of documents returned by a standard search engine (e.g. VSR) be called the *root* set *R*.
- Initialize the base set *S* to *R*.
- Add to *S* all pages pointed to by any page in *R*.
- Add to *S* all pages that point to any page in *R*.



Neighborhood Graph

- Subgraph associated to each query



Base Limitations

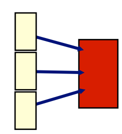
- To limit computational expense:
  - Limit number of root pages to the top 200 pages retrieved for the query.
  - Limit number of "back-pointer" pages to a random set of at most 50 pages returned by a "reverse link" query.
- To eliminate purely navigational links:
  - Eliminate links between two pages on the same host.
- To eliminate "non-authority-conveying" links:
  - Allow only *m* (*m* ≅ 4-8) pages from a given host as pointers to any individual page.

Authorities and In-Degree

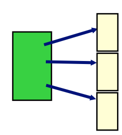
- Even within the base set *S* for a given query, the nodes with highest in-degree are not necessarily authorities (may just be generally popular pages like Yahoo or Amazon).
- True authority pages are pointed to by a number of hubs (i.e., pages that point to lots of authorities).

HITS

- Goal: Given a query find:
  - Good sources of content (authorities)

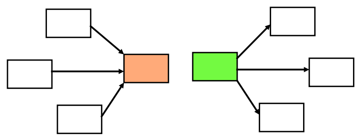


  - Good sources of links (hubs)

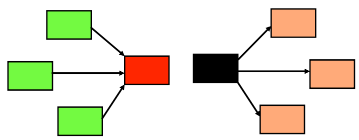


Intuition

- Authority **comes from** in-edges.
  Being a good hub **comes from** out-edges.



- Better authority **comes from** in-edges from good hubs. Being a better hub **comes from** out-edges to good authorities.



Iterative Algorithm

- Use an iterative algorithm to slowly converge on a mutually reinforcing set of hubs and authorities.
- Maintain for each page $p \in S$:
  - Authority score: a_p (vector $\boldsymbol{a}$)
  - Hub score: h_p (vector $\boldsymbol{h}$)
- Initialize all $a_p = h_p = 1$
- Maintain normalized scores:

$$\sum_{p \in S} (a_p)^2 = 1 \qquad \sum_{p \in S} (h_p)^2 = 1$$

- Repeat until vectors *a and h converge.*

HITS Iterative Algorithm

Initialize for all $p \in S$: $a_p = h_p = 1$

For i = 1 to k:

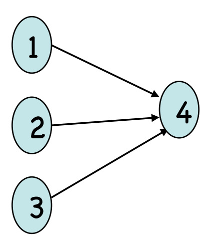
For all $p \in S$: $a_p = \sum_{q: q \rightarrow p} h_q$ (*update auth. scores*)

For all $p \in S$: $h_p = \sum_{q: p \rightarrow q} a_q$ (*update hub scores*)

For all $p \in S$: $a_p = a_p/c$ c: $\sum_{p \in S} (a_p / c)^2 = 1$ (*normalize* ***a***)

For all $p \in S$: $h_p = h_p/c$ c: $\sum_{p \in S} (h_p / c)^2 = 1$ (*normalize* ***h***)

Illustrated Update Rules



$a_4 = h_1 + h_2 + h_3$

$h_4 = a_5 + a_6 + a_7$