Searching the Web [Arasu 01]

- Most user simply browse the web
  - Google, Yahoo, Lycos, Ask...
- Others do more specialized searches
  - web search engines
  - submit queries by specifying lists of keywords
  - receive web pages containing the keywords
Several studies estimated the size of the web:

- over a billion pages
- assuming average size of web page 5-10 KB
- Size of web is tens of terabytes
- the size of the web has doubled in less than two years
- this growth rate is expected to continue
Structure of the Web

- The link structure of the web is like a 'bow tie'
  - Broder et al. 2000
  - connected core: 28%
  - 'IN' loop: 22% pages which can reach the core
  - 'OUT' loop: 22% pages that can be reached from the core
  - remaining nodes are isolated
IR on the Web

- **IR**: best suited for small – coherent collections
  - digital libraries, books, journal articles, newspapers

- **Web**: massive, less coherent, distributed, continuously updating information, not all pages are important
  - IR alone is not sufficient
  - exploit linkage among web pages
Web Search Engine

Arasu 01
Web Crawler

- Small programs that retrieve pages from the web for later analysis by the indexing module
  - follow links between pages
  - pass URLs to crawler control module to determine which pages to visit next
  - pass retrieved pages into page repository
  - continue visiting pages until local resources are exhausted
Algorithm

- Starts with an initial set of URLs
- Repeat until it decides to stop
  - place URLs in a queue
  - prioritize URLs in the queue
  - get next URL and downloads the page
- **Larbin**: a crawler for your PC!
Arising Issues

- What pages to download
  - cannot download all pages
  - visit important pages by prioritizing URLs in queue
- Refresh pages
  - revisit downloaded pages and detect changes
  - some pages change more frequently than others
- Minimize load on visited sites
  - consumes resources belonging to others
- Parallelize processing
  - crawlers run on multiple machines
  - download pages in parallel
Page Refreshing

- Maintain crawled pages up-to-date
  - **uniform refresh policy**: revisit all pages at frequency $f$
  - **proportional refresh policy**: revisit pages that change more often
- Difficult to keep track of pages that change often
- Do not visit such pages very frequently
- Uniform refresh policy is better
Page Selection

- Download important pages first
- Importance measures can be defined in many ways
  - interest driven: textual similarity
  - popularity driven: back-link count of page
  - location driven: endings (.com), fewer slashes ...
  - combination of the above or defined based on link analysis (e.g., PageRank)
Crawler Models \[\text{Aggrawal 01}\]

- **Crawl and stop**: stop after visiting $K$ pages
  - only $M$ pages have rank less than $r$ (num. of pages that an ideal crawler would produce)
  - performance: $M/K$

- **Crawl and stop with threshold**: visit $K$ pages but also apply an acceptance criterion
  - qualifying or hot pages $H$
  - performance: $H/K$
Performance of Crawler Models

Arasu'01

[Graph showing performance of different crawler models with hot pages crawled on the y-axis and pages crawled on the x-axis. The graph compares PageRank, backlink, breadth, and random ordering metrics.]
Storage

- Large database, similar to data base systems but, different functionality
  - less transaction management, logging ...

- The focus is on
  - scalability: large distributed data repository
  - dual access modes: random (specific page), streaming access (set of pages)
  - updates: high rate of modifications (delete or update obsolete pages)
Indexing

- **Indexer**: extracts URLs and terms from pages, creates a lookup table of URLs
- **Text index**: inverted file, points to pages where a given term occurs
- **Structure index**: based on links between pages, e.g., backward, forward links per page
- **Utility index**: created by the collection analysis module, more specialized indexes based on importance, image content etc.
Indexing

- **Index partitioning**: distributed index
- **Local inverted file**: nodes storing disjoint subsets of pages
  - Each query addresses a separate index
- **Global inverted file**: each node indexes part of the terms e.g., [e-f],[f-h]...
  - Each query addresses multiple indices in parallel
Web Indexing Architecture

Arasu’01

break query or page in multiple “runs”
Ranking Search Results

- Traditional IR is not very effective in ranking query results
  - web is huge
  - great variation of content
  - pages that contain the query terms may be irrelevant or unimportant
  - small queries (1-2 terms) return huge answer sets

- Solution: link analysis
Link Analysis

- The link structure of the web is important for filtering irrelevant or unimportant pages.

- Main idea: if $p \rightarrow q$ then
  - $p$, $q$ are related
  - $p$ recommends $q$

- Two main approaches: HITS & PageRank
  - for each page compute a value that captures the notion of importance of the page.
PageRank

- Measure of page importance
- Global ranking scheme, computes a single rank to every page
- Google's page is more important than my page
- The difference is reflected in the number and quality of pages pointing to each page
- A page is important if other important pages are pointing to it
- Recursive definition
Simple PageRank

- Let the pages of the web be 1, 2, ..., m
  - $N_j$: number of forward links
  - $B_j$: number of backward links
  - $r_i$: PageRank of page $i$
- The pages that point to $i$ distribute their rank to the pages they point to

$$r_i = \sum_{j \in B_i} \frac{r_j}{N_j}$$
Example of PageRank

- $r_2 = 0.286$ which is distributed to $r_1, r_3$
- $r_3 = 0.286/2$ since it has no other incoming links
- $r_1 = 0.286$ the rank it receives from $r_5, r_2$ and $r_3$
- $\sum r_i = 1$
Computing PageRank

- The PageRank of page $i$ is the eigenvector of $A^T$ corresponding to eigenvalue 1

$$r_i = \sum_{j \in B_i} \frac{r_j}{N_j} \iff r = A^T r$$

where

$$r = [r_1 \, r_2 \ldots r_m]$$

$$a_{ij} = \begin{cases} 1/N_i & \text{if } i \text{ points to } j, i \text{ points to } N_i \text{ pages} \\ 0 & \text{otherwise} \end{cases}$$
Computing PageRank (cont,d)

- **Power iteration** method for computing the principal eigenvector of a matrix
- An arbitrary initial vector is multiplied repeatedly with $A^T$ until it converges to the principal eigenvector

1. $s = \text{random vector}$
2. $r = A^T s$
3. if ($|r - s| < e$) stop : $r$ is the PageRank vector
   
   else  
   
   \{$s = r; \text{goto step 2}$\}
Random Surfer Model

- Definition of PageRank leads itself to an interpretation based on “random walks” [Motwani, Raghavan 1995]
- A person who surfs the web by randomly clicking links does a random walk on the web
- The PageRank of a page is proportional to the frequency with which a random surfer would visit it
Practical PageRank

- Well defined for strongly connected graphs
- **sink**: connected pages with no links pointing outwards
- nodes not in the sink receive 0 PageRank
- **leak**: a page with no outlinks
  - e.g., 5 becomes a leak if 5→4 is removed
- **sinks and leaks cause rank** → 0 for nodes not in sinks or leaks
Leaks and Sinks

- A random surfer would eventually get stuck in nodes 4 and 5
  - nodes 1, 2, and 3 would have rank 0 and nodes 4 and 5 would have rank 0.5

- By reaching a rank leak he is lost forever
  - a leak causes all the ranks to eventually converge to 0

- Our random surfer would eventually reach node 4 and will never be seen again!
Computing PageRank

- **Leaks**: Eliminate all nodes with no out degree or assume they have links to the pages that point to them
- **Sinks**: Assume arbitrary links to the outside connected graph
- Compute page rank using Random Surfer model [Page, 1998]
  - Whenever a surfer gets trapped in a sink will jump out to a random page after a number of loops
Assume that only a fraction $d$ ($0 < d < 1$) of the rank of a page is distributed among the pages it points to.

The remaining rank is distributed equally among all the $m$ pages on the web.

Parameter $d$ dictates how often the surfer gets bored (random surfer model).

$$r_i = d \sum_{j \in N_j} \frac{r_j}{N_j} + \frac{1-d}{m}$$
Google [Brin 98]

- First prototype at Stanford University
- IR score is combined with PageRank to determine the rank of an answer
  - given a query, Google computes the IR score (e.g., vector model) of pages containing the query terms
- Google also uses more text-based techniques to enhance results (e.g., anchor text on links)
Comparison

Nodes 4 and 5 now have higher ranks than the other nodes, indicating that surfers will tend to gravitate to 4 and 5. However, the other nodes have nonzero ranks.
HITS [Kleinberg 98]

- HITS is a query dependent ranking technique
  - authority and hub score to each page
- Authorities: pages which are likely to be relevant to the query
- Hubs: pages pointing to authorities
  - some hubs may be authorities too
  - used to compute the authorities
- Good authorities are pointed to by many hubs
- Good hubs point to many authorities
Hubs

- The *hub* pages are not necessarily authorities themselves but point to several authority pages.
- There are two reasons why one might be interested in hubs pages:
  - They are used to compute the authority pages.
  - They can also be returned to the user in response to a query.
Focused Subgraph F

- HITS does link analysis on $F$:
  1) compute set $R$ of pages relevant to the query (containing the query terms)
  2) initially $F = R$ (accept maximum $t$ pages)
  3) for each page in $p$ in $R$
     - include in $F$ all pages that $p$ points to
     - include pages pointing to $p$ (maximum $d$ pages)

- $t$, $d$ are parameters, defined by designer
hubs

authorities
HITS Link Analysis

- $F = \{1, 2\ldots N\}$ pages
  - $F_i$ number of pages that $i$ points to
  - $B_i$ number of pages pointing to $i$
- $A$ adjacency matrix:
  - $a_{ij} = 1$ if $i$ points to $j$,
  - $a_{ij} = 0$ otherwise
Computing HITS

- For each $i$ compute
  - **authority score** $a_i$: sum of hub scores of pages pointing to it
  - **hub score** $h_i$: sum of authority scores of all pages it points to
- $h, a$: principal eigenvectors of $AA^T, A^TA$

\[
\begin{align*}
a_i &= \sum_{j \in B_i} h_j \iff a = A^T h \iff h = AA^T h \\
h_i &= \sum_{j \in F_i} a_j \iff h = Aa \iff a = A^T Aa
\end{align*}
\]
Bibliographic Matrix

- The two matrices $A^T A$ and $AA^T$ are well known in the field of bibliometrics:
  - $A^T A$ is the \textit{co-citation matrix} of the collection
    - $[A^T A]_{i,j}$: number of pages which jointly point at (cite) pages $i$ and $j$
  - $AA^T$ is the \textit{bibliographic coupling matrix} of the collection
    - $[AA^T]_{i,j}$: number of pages jointly referred to (pointed at) by pages $i$ and $j$
Computing HITS

- Power iteration (similarly to PageRank)
- Initialize $a_i$, $h_i$ to arbitrary values
- Repeat until convergence
  1. $a_i = \sum_{j \in B_i} h_j$
  2. $h_i = \sum_{j \in F_i} a_j$
  3. At each step normalize each $a_i$, $h_i$ by
     $\{\sum_i a_i^2\}^{1/2}$, $\{\sum_i h_i^2\}^{1/2}$
Example

\[
\begin{align*}
h &= 0 & h &= 0 & h &= 0 \\
a &= 0.408 & a &= 0.816 & a &= 0.408 \\
4 & & 5 & & 6 \\
\end{align*}
\]

\[
\begin{align*}
1 & & 2 & & 3 \\
h &= 0.408 & h &= 0.816 & h &= 0.408 \\
a &= 0 & a &= 0 & a &= 0 \\
\end{align*}
\]
Multimedia and Link Analysis

- PageRank and HITS have been investigated in conjunction with text retrieval techniques
- do the same for images and video
- PicASHOW extends HITS for images
- Diogenis is a web system for person images
- ImageRover, PicToSeek, WebSeek: web systems for images
Data Mining on Web

- Link analysis in conjunction with text, image data to perform *data mining* on the web (e.g., Chackrabarti 99)
- Identify *communities* on the web (e.g., Gibson 98)
- Define and produce *ontologies* similar to Yahoo’s topic taxonomies
- Find related pages
Image Retrieval on the Web using Link Analysis

- Queries: keywords
- Images are described by text surrounding them in Web pages
  - image filename, Alternate text, Page title, Caption
- Boolean, VSM.. to retrieval Web pages
- PageRank, HITs to assign higher ranking to images in important Web pages
- Important Web page does not mean relevant images!
Image Importance

- **Answers**: images in Web pages
  - Relevant but not always important
  - From corporate web sites, organizations
  - From individuals and small companies

- **Link analysis**: assign higher ranking to answers from important web sites
PicASHOW [Lempel 2002]

- PageRank and HITS for retrieval using keywords
- PicASHOW for Web pages with images
- Main idea: co-cited and co-contained images are likely to be related
  - W: page to page relationships
  - M: page to image relationships
Example
Image Link Analysis

- Queries are matched against text descriptions
- Create the focused sub-graph $F$ of the initial answer set
- Authorities: principal eigenvector of $[(W+I)M^T](W+I)M$
  - $W$: page to page relationships in $F$
  - $M$: page to image relationships in $F$
- Rank answers by authority value
Further Reading


