This lecture
- Wrap up pagerank
- Anchor text
- HITS
- Behavioral ranking

Pagerank: Issues and Variants
- 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.
- Biased Surfer Models
  - Weight edge traversal probabilities based on match with topic/query (non-uniform edge selection)
  - Bias jumps to pages on topic (e.g., based on personal bookmarks & categories of interest)

Topic Specific Pagerank [Have02]
- 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 & user–specific distribution over the categories
  - 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 [Have02]
- Implementation
  - offline: Compute pagerank distributions wrt to 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
    Generate a dynamic pagerank score for each page – weighted sum of category-specific pageranks

Influencing PageRank (“Personalization”)
- Input:
  - Web graph \( W \)
  - Influence vector \( v \)
    \( v : \text{(page} \rightarrow \text{degree of influence)} \)
- Output:
  - Rank vector \( r \) (page \( \rightarrow \) page importance wrt \( v \))
  - \( r = \text{PR}(W, v) \)
Non-uniform Teleportation
Teleport with 10% probability to a Sports page

Interpretation of Composite Score
- For a set of personalization vectors \( \{v_j\} \)
  \[ \sum_j [w_j \cdot PR(W, v_j)] = PR(W, \sum_j [w_j \cdot v_j]) \]
- Weighted sum of rank vectors itself forms a valid rank vector, because PR() is linear wrt \( v_j \)

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

The Web as a Directed Graph
Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)
Assumption 2: The anchor of the hyperlink describes the target page (textual context)
Assumptions Tested

- A link is an endorsement (quality signal)
  - Except when affiliated
- Can we recognize affiliated links? [Davi00]
  - 1536 links manually labeled
  - 59 binary features (e.g., on-domain, meta tag overlap, common outlinks)
  - C4.5 decision tree, 10 fold cross validation showed 98.7% accuracy
    - Additional surrounding text has lower probability but can be useful

Assumptions tested

- Anchors describe the target
  - Topical Locality [Davi00b]
  - ~200K pages (query results + their outlinks)
  - Computed “page to page” similarity (TFIDF measure)
    - Link-to-Same-Domain > Cocited > Link-to-Different-Domain
  - Computed “anchor to page” similarity
    - Mean anchor len = 2.69
    - 0.6 mean probability of an anchor term in target page

Anchor Text

WWW Worm – McBryan [Mcbr94]

- For [ibm] how to distinguish between:
  - IBM’s home page (mostly graphical)
  - IBM’s copyright page (high term freq. for ‘ibm’)
  - Rival’s spam page (arbitrarily high term freq.)

A million pieces of anchor text with “ibm” send a strong signal

- “ibm” to “ibm.com” to “IBM home page”

Indexing anchor text

- When indexing a document $D$, include anchor text from links pointing to $D$.

Indexing anchor text

- Can sometimes have unexpected side effects – e.g., evil empire.
- Can index anchor text with less weight.

Anchor Text

- Other applications
  - Weighting/filtering links in the graph
    - HITS [Chak98], Hilltop [Bhar01]
  - Generating page descriptions from anchor text [Amit98, Amit00]
Hyperlink-Induced Topic Search (HITS) – Klei98

- In response to a query, instead of an ordered list of pages each meeting the query, find two sets of inter-related pages:
  - *Hub pages* are good lists of links on a subject.
    - e.g., "Bob's list of cancer-related links."
  - *Authority pages* occur recurrently on good hubs for the subject.
- Best suited for "broad topic" queries rather than for page-finding queries.
- Gets at a broader slice of common opinion.

Hubs and Authorities

- Thus, a good hub page for a topic *points* to many authoritative pages for that topic.
- A good authority page for a topic is *pointed* to by many good hubs for that topic.
- Circular definition – will turn this into an iterative computation.

The hope

```
Alice   AT&T
|       |
|       |
|       |
```

```
Bob     Sprint
|       |
|       |
|       |
```

```
MCI
```

Long distance telephone companies

High-level scheme

- Extract from the web a base set of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages; → iterative algorithm.

Base set

- Given text query (say *browser*), use a text index to get all pages containing *browser*.
  - Call this the *root set* of pages.
- Add in any page that either
  - points to a page in the root set, or
  - is pointed to by a page in the root set.
- Call this the *base set*.

Visualization
Assembling the base set [Klei98]

- Root set typically 200–1000 nodes.
- Base set may have up to 5000 nodes.
- How do you find the base set nodes?
  - Follow out-links by parsing root set pages.
  - Get in-links (and out-links) from a connectivity server.
  - (Actually, suffices to text-index strings of the form \texttt{href=URL} to get in-links to \texttt{URL}.)

Distilling hubs and authorities

- Compute, for each page \(x\) in the base set, a hub score \(h(x)\) and an authority score \(a(x)\).
- Initialize: for all \(x\), \(h(x) \leftarrow 1; a(x) \leftarrow 1\).
- Iteratively update all \(h(x), a(x)\);
- After iterations
  - output pages with highest \(h()\) scores as top hubs
  - highest \(a()\) scores as top authorities.

Iterative update

- Repeat the following updates, for all \(x\):
  \[
  h(x) \leftarrow \sum_{y \rightarrow x} a(y)
  \]
  \[
  a(x) \leftarrow \sum_{y \leftarrow x} h(y)
  \]

Scaling

- To prevent the \(h()\) and \(a()\) values from getting too big, can scale down after each iteration.
- Scaling factor doesn’t really matter:
  - we only care about the relative values of the scores.

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
  - in fact, suitably scaled, \(h()\) and \(a()\) scores settle into a steady state!
  - proof of this comes later.
- We only require the relative orders of the \(h()\) and \(a()\) scores – not their absolute values.
- In practice, \(~5\) iterations get you close to stability.

Japan Elementary Schools

<table>
<thead>
<tr>
<th>Hubs</th>
<th>Authorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>school</td>
<td>The American School in Japan</td>
</tr>
<tr>
<td>LINK Page-13</td>
<td>The LMS Page</td>
</tr>
<tr>
<td>u0.5[1]_Ver 2</td>
<td>u0.5[1]_Ver 2</td>
</tr>
<tr>
<td>a0.5—Ver 2324[1]_Ver 2</td>
<td></td>
</tr>
<tr>
<td>100 Schools HomePage</td>
<td>University</td>
</tr>
<tr>
<td>(Eng)</td>
<td></td>
</tr>
<tr>
<td>K-12 from Japan</td>
<td></td>
</tr>
<tr>
<td>toshie.net</td>
<td></td>
</tr>
<tr>
<td><a href="http://www">http://www</a>...</td>
<td></td>
</tr>
<tr>
<td>-site inactive</td>
<td></td>
</tr>
<tr>
<td>-site inactive</td>
<td></td>
</tr>
<tr>
<td>-site inactive</td>
<td></td>
</tr>
<tr>
<td>kidspeace</td>
<td></td>
</tr>
<tr>
<td>Koulutus ja oppilaitokset</td>
<td></td>
</tr>
<tr>
<td>TOYOSA HOMEPAGE</td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td></td>
</tr>
<tr>
<td>Cay’s Homepage(Japanese)</td>
<td></td>
</tr>
<tr>
<td>œ*1—Ver 2324[1]_Ver 2</td>
<td></td>
</tr>
<tr>
<td>UNIVERSITY</td>
<td></td>
</tr>
<tr>
<td>œ*1—Ver 2324[1]_Ver 2</td>
<td></td>
</tr>
<tr>
<td>UNIVERSITY</td>
<td></td>
</tr>
<tr>
<td>œ*1—Ver 2324[1]_Ver 2</td>
<td></td>
</tr>
<tr>
<td>œ*1—Ver 2324[1]_Ver 2</td>
<td></td>
</tr>
<tr>
<td>œ*1—Ver 2324[1]_Ver 2</td>
<td></td>
</tr>
</tbody>
</table>
Things to note

- Pulled together good pages regardless of language of page content.
- Use only link analysis after base set assembled
  - Iterative scoring is query-independent.
- Iterative computation after text index retrieval — significant overhead.

Proof of convergence

- n x n adjacency matrix A:
  - each of the n pages in the base set has a row and column in the matrix.
  - Entry A_{ij} = 1 if page i links to page j, else = 0.

\[
\begin{pmatrix}
1 & 0 & 1 \\
2 & 1 & 1 \\
3 & 1 & 0 \\
\end{pmatrix}
\]

Hub/authority vectors

- View the hub scores h() and the authority scores a() as vectors with n components.
- Recall the iterative updates
  \[
  h(x) \leftarrow \sum_{y \rightarrow x} a(y)
  \]
  \[
  a(x) \leftarrow \sum_{x \rightarrow y} h(y)
  \]

Rewrite in matrix form

- h = Aa.
- a = A^T h.  

Substituting, h = AA^T h and a = A^T Aa.  
Thus, h is an eigenvector of AA^T and a is an eigenvector of A^T A.

Further, our algorithm is a particular, known algorithm for computing eigenvectors: the power iteration method.

Solutions

- ARC [Chak98] and Clever [Chak98b]
  - Distance-2 neighborhood graph
  - Tackling affiliated linkage
    - IP prefix (e.g., 208.47.*) rather than hosts to identify “same author” pages
  - Tackling topic drift
    - Weight edges by match between query and extended anchor text
    - Distribute hub score non-uniformly to outlinks
  - Intuition: Regions of the hub page with links to good authorities get more of the hub score

(For follow-up based on Document Object Model see [Chak01])
Solutions (contd)

- Topic Distillation [Bhar98]
  - Tackling affiliated linkage
  - Normalize weights of edges from/to a single host
  - Tackling topic drift
  - Query expansion.
  - “Topic vector” computed from docs in the initial ranking.
  - Match with topic vector used to weight edges and remove off-topic nodes
- Evaluation
  - 28 broad queries. Pooled results, blind ratings of results by 3 reviewers per query
  - Average precision @ 10
    - Topic Distillation = 0.66, HITS = 0.46

Hilltop
[Bhar01]

- Preprocessing: Special index of “expert” hubs
  - Select a subset of the web (~ 5%)
  - High out-degree to non-affiliated pages on a theme
- At query time compute:
  - Expert score (Hub score)
    - Based on text match between query and expert hub
  - Authority score
    - Based on scores of non-affiliated experts pointing to the given page
    - Also based on match between query and extended anchor-text (includes enclosing headings + title)
  - Return top ranked pages by authority score

Behavior–based ranking

- For each query \( Q \), keep track of which docs in the results are clicked on
- On subsequent requests for \( Q \), re-order docs in results based on click–throughs
- First due to DirectHit \( \rightarrow \) AskJeeves
- Relevance assessment based on
  - Behavior/usage
  - vs. content

Query–doc popularity matrix \( B \)

\[
B_{qj} = \text{number of times doc } j \text{ clicked-through on query } q
\]

When query \( q \) issued again, order docs by \( B_{qj} \) values.

Issues to consider

- Weighing/combining text- and click-based scores.
- What identifies a query?
  - Ferrari Mondial
  - Ferrari Mondial
  - Ferrari mondial
  - ferrari mondial
  - “Ferrari Mondial”
- Can use heuristics, but search parsing slowed.
Vector space implementation

- Maintain a term-doc popularity matrix $C$
  - as opposed to query-doc popularity
  - initialized to all zeros
- Each column represents a doc $j$
  - If doc $j$ clicked on for query $q$, update $C_j \leftarrow C_j + \varepsilon q$ (here $q$ is viewed as a vector).
- On a query $q'$, compute its cosine proximity to $C_j$ for all $j$.
- Combine this with the regular text score.

Issues

- Normalization of $C_j$ after updating
- Assumption of query compositionality
  - “white house” document popularity derived from “white” and “house”
- Updating – live or batch?

Basic Assumption

- Relevance can be directly measured by number of click throughs
- Valid?

Validity of Basic Assumption

- Click through to docs that turn out to be non-relevant: what does a click mean?
- Self-perpetuating ranking
- Spam
- All votes count the same

Variants

- Time spent viewing page
  - Difficult session management
  - Inconclusive modeling so far
- Does user back out of page?
- Does user stop searching?
- Does user transact?