Web Crawling and Data Gathering

Spidering

Some Typical Tasks

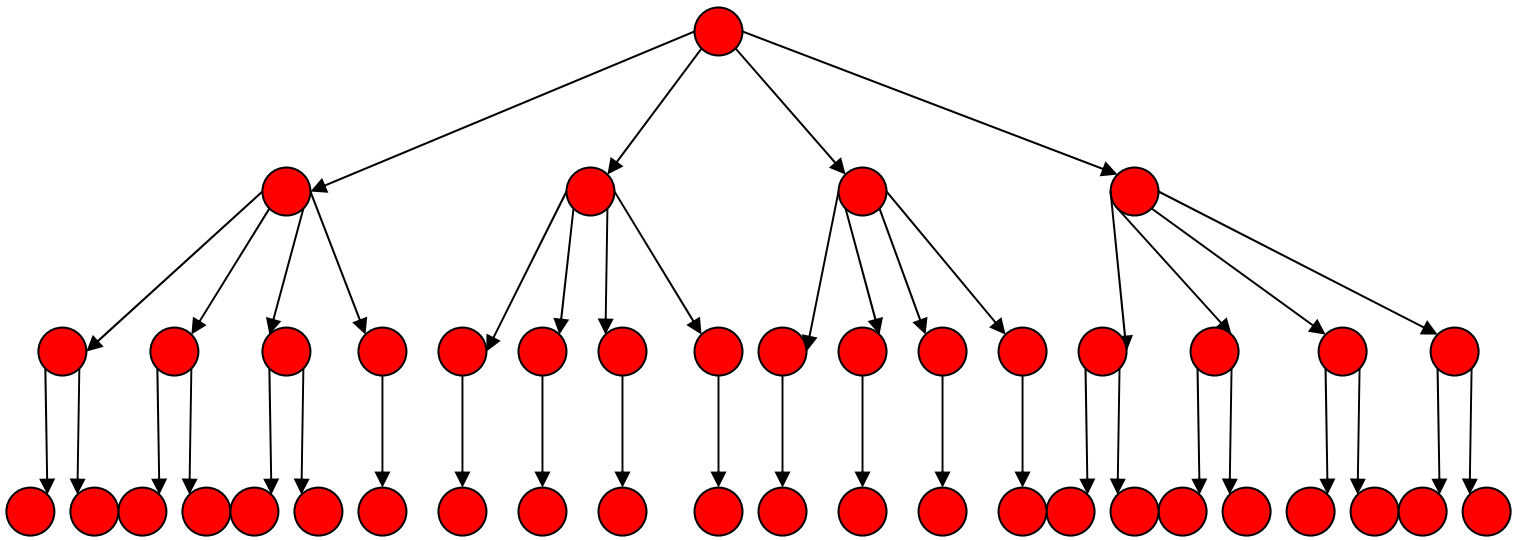
- Get information from other parts of an organization
  - It may be easier to get information yourself than to ask others to give it to you
- Get information from external sites
  - Competitors, media, user groups
- Build a corpus of documents for a particular task
  - Home pages, educational sites, shopping sites
- Build a corpus for text mining
  - Large, representative of some domain

Spiders (Robots/Bots/Crawlers)

- Start with a comprehensive set of root URL's from which to start the search.
- Follow all links on these pages recursively to find additional pages.
- Index all **novel** found pages in an inverted index as they are encountered.
- May allow users to directly submit pages to be indexed (and crawled from).

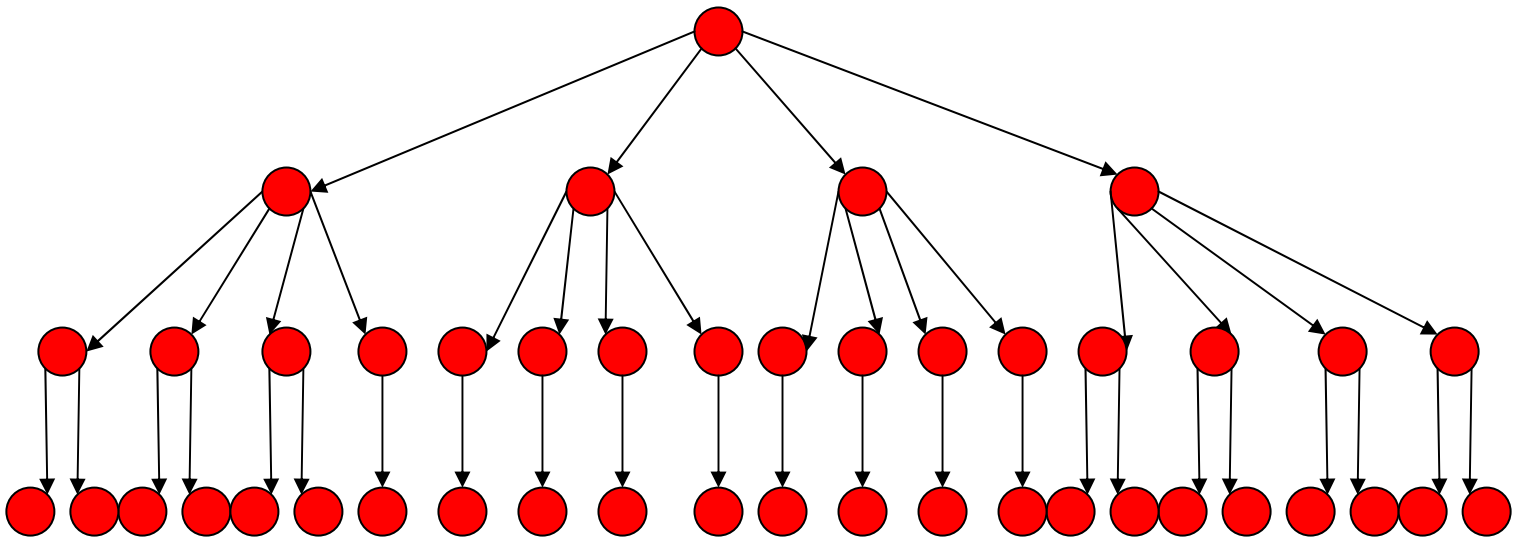
Search Strategies

Breadth-first Search



Search Strategies (cont)

Depth-first Search



Search Strategy Trade-Off's

- Breadth-first explores uniformly outward from the root page but requires memory of all nodes on the previous level (exponential in depth). Standard spidering method. Queue the links.
- Depth-first requires memory of only depth times branching-factor (linear in depth) but gets "lost" pursuing a single thread. Stack the links.
- Both strategies implementable using a list of links (URL's).

Avoiding Page Duplication

- Must detect when revisiting a page that has already been spidered (web is a graph not a tree).
- Must efficiently index visited pages to allow rapid recognition test.
  - Tree indexing (e.g. trie)
  - Hashtable
- Index page using URL as a key.
  - Must canonicalize URL's (e.g. delete ending "/")
  - Not detect duplicated or mirrored pages.
- Index page using textual content as a key.
  - Requires first downloading page.

A Description of the Activities

1. Initialize a page queue with one or a few known sites
   1. E.g. http://www.cs.umass.edu
2. Pop an address from the queue
3. Get the page
4. Parse the page to find other URLs
   1. E.g. Recent News
5. Discard URLs that do not meet requirements
   1. E.g. images, executables, PostScript, PDF, zip …
   2. Eg. Pages that have been seen before
6. Add the URLs to the queue
7. If not time to stop, go to step 2

Spidering Algorithm

```
Initialize queue (Q) with initial set of known URL's.
Until Q empty or page or time limit exhausted:
    Pop URL, L, from front of Q.
    If L is not to an HTML page (.gif, .jpeg, .ps, .pdf, .ppt…)
        continue loop.
    If already visited L, continue loop.
    Download page, P, for L.
    If cannot download P (e.g. 404 error, robot excluded)
        continue loop.
    Index P (e.g. add to inverted index or store cached copy).
    Parse P to obtain list of new links N.
    Append N to the end of Q.
```

Insert Slides from Umass Notes

- A simple spider architecture
- A simple spider architecture -- main process
- A simple spider architecture -- crawler process and downloading threads
- A simple architecture -- characteristics

Queueing Strategy

- How new links added to the queue determines search strategy.
- FIFO (append to end of Q) gives breadth-first search.
- LIFO (add to front of Q) gives depth-first search.
- Heuristically ordering the Q (priority queue) gives a “focused crawler” that directs its search towards “interesting” pages.

Restricting Spidering

- Restrict spider to a particular site.
  - Remove links to other sites from Q.
- Restrict spider to a particular directory.
  - Remove links not in the specified directory.
- Obey page-owner restrictions (robot exclusion).

Link Extraction

- Must find all links in a page and extract URLs.
  -
  - <frame src="site-index.html">
- Must complete relative URL's using current page URL:
  - to http://www.cs.utexas.edu/users/mooney/ir-course/proj3
  - to http://www.cs.utexas.edu/users/mooney/cs343/syllabus.html

URL Syntax

- A URL has the following syntax:
  - <scheme>://<authority><path>?<query>#<fragment>
- An *authority* has the syntax:
  - <host>:<port-number>
- A *query* passes variable values from an HTML form and has the syntax:
  - <variable>=<value>&<variable>=<value>…
- A *fragment* is also called a *reference* or a *ref* and is a pointer within the document to a point specified by an anchor tag of the form:
  - <A NAME="<fragment>">

Link Canonicalization

- Equivalent variations of ending directory normalized by removing ending slash.
  - http://www.cs.utexas.edu/users/mooney/
  - http://www.cs.utexas.edu/users/mooney
- Internal page fragments (ref's) removed:
  - http://www.cs.utexas.edu/users/mooney/welcome.html#courses
  - http://www.cs.utexas.edu/users/mooney/welcome.html

Anchor Text Indexing

- Extract anchor text (between <a> and) of each link followed.
- Anchor text is usually descriptive of the document to which it points.
- Add anchor text to the content of the destination page to provide additional relevant keyword indices.
- Used by Google:
  - Evil Empire
  - IBM

Anchor Text Indexing (cont)

- Helps when descriptive text in destination page is embedded in image logos rather than in accessible text.
- Many times anchor text is not useful:
  - “click here”
- Increases content more for popular pages with many in-coming links, increasing recall of these pages.
- May even give higher weights to tokens from anchor text.

Robot Exclusion

- Web sites and pages can specify that robots should not crawl/index certain areas.
- Two components:
  - Robots Exclusion Protocol: Site wide specification of excluded directories.
  - Robots META Tag: Individual document tag to exclude indexing or following links.
- Http://www.robotstxt.org/wc/exclusion.html

Robots Exclusion Protocol

- Site administrator puts a “robots.txt” file at the root of the host’s web directory.
  - http://www.ebay.com/robots.txt
  - http://www.abcnews.com/robots.txt
- File is a list of excluded directories for a given robot (user-agent).
  - Exclude all robots from the entire site:

```
User-agent: *
Disallow: /
```

Robot Exclusion Protocol Examples

- Exclude specific directories:

```
User-agent: *
Disallow: /tmp/
Disallow: /cgi-bin/
Disallow: /users/paranoid/
```

- Exclude a specific robot:

```
User-agent: GoogleBot
Disallow: /
```

- Allow a specific robot:

```
User-agent: GoogleBot
Disallow:

User-agent: *
Disallow: /
```

Robot Exclusion Protocol Details

- Only use blank lines to separate different User-agent disallowed directories.
- One directory per “Disallow” line.
- No regex patterns in directories.

Robots META Tag

- Include META tag in HEAD section of a specific HTML document.
  - <meta name="robots" content="none">
- Content value is a pair of values for two aspects:
  - index | noindex: Allow/disallow indexing of this page.
  - follow | nofollow: Allow/disallow following links on this page.

Robots META Tag (cont)

- Special values:
  - all = index,follow
  - none = noindex,nofollow
- Examples:

```
<meta name="robots" content="noindex,follow">
<meta name="robots" content="index,nofollow">
<meta name="robots" content="none">
```

Robot Exclusion Issues

- META tag is newer and less well-adopted than "robots.txt".
- Standards are conventions to be followed by "good robots."
- Companies have been prosecuted for "disobeying" these conventions and "trespassing" on private cyberspace.
- "Good robots" also try not to "hammer" individual sites with lots of rapid requests.
  - "Denial of service" attack.

Good Behavior

- Wait 5 minutes between downloads from a particular server
- Self-identification via the User-Agent field
  - Spider name, email address, project URL …
  - Web site administrators are less likely to block access if they can see who is running the spider and why

Multi-Threaded Spidering

- Bottleneck is network delay in downloading individual pages.
- Best to have multiple threads running in parallel each requesting a page from a different host.
- Distribute URL's to threads to guarantee equitable distribution of requests across different hosts to maximize through-put and avoid overloading any single server.
- Early Google spider had multiple co-ordinated crawlers with about 300 threads each, together able to download over 100 pages per second.

Directed/Focused Spidering

- Sort queue to explore more “interesting” pages first.
- Two styles of focus:
  - Topic-Directed
  - Link-Directed

Topic-Directed Spidering

- Assume desired topic description or sample pages of interest are given.
- Sort queue of links by the similarity (e.g. cosine metric) of their source pages and/or anchor text to this topic description.
- Preferentially explores pages related to a specific topic.

Link-Directed Spidering

- Monitor links and keep track of in-degree and out-degree of each page encountered.
- Sort queue to prefer popular pages with many in-coming links (*authorities*).
- Sort queue to prefer summary pages with many out-going links (*hubs*).

Keeping Spidered Pages Up to Date

- Web is very dynamic: many new pages, updated pages, deleted pages, etc.
- Periodically check spidered pages for updates and deletions:
  - Just look at header info (e.g. META tags on last update) to determine if page has changed, only reload entire page if needed.
- Track how often each page is updated and preferentially return to pages which are historically more dynamic.
- Preferentially update pages that are accessed more often to optimize freshness of more popular pages.

Gathering Data: Common Problems

- Relative paths
  - Eg: Hamlet
- Frames: follow links in frame tags
- Cycles: same page, different addresses
- Black holes: “Next year” links on a calender page
- Scripts: HTML written dynamically
- Non-conforming pages: buggy HTML pages
- Flaky communications: slow links, partial downloads
- Huge pages: what is a limit? 10 MB page?

For More Information

- Martijn Koster, “Guidelines for Robot Writers”, 1993
  - http://www.robotstxt.org/wc/guidelines.html
- J. Cho, H. Garcia-Molina, L. Page, “Efficient crawling through URL ordering”, In *Proceedings of the 7th World Wide Web Conference*. 1998
  - http://dbpubs.stanford.edu/pub/1998-51
- S. Brin and L. Page “The anatomy of a large-scale hypertextual web search engine”, in *Proceedings of the 7th World Wide Web Conference*. 1998
  - http://dbpubs.stanford.edu:8090/pub/1998-8