
Web Search

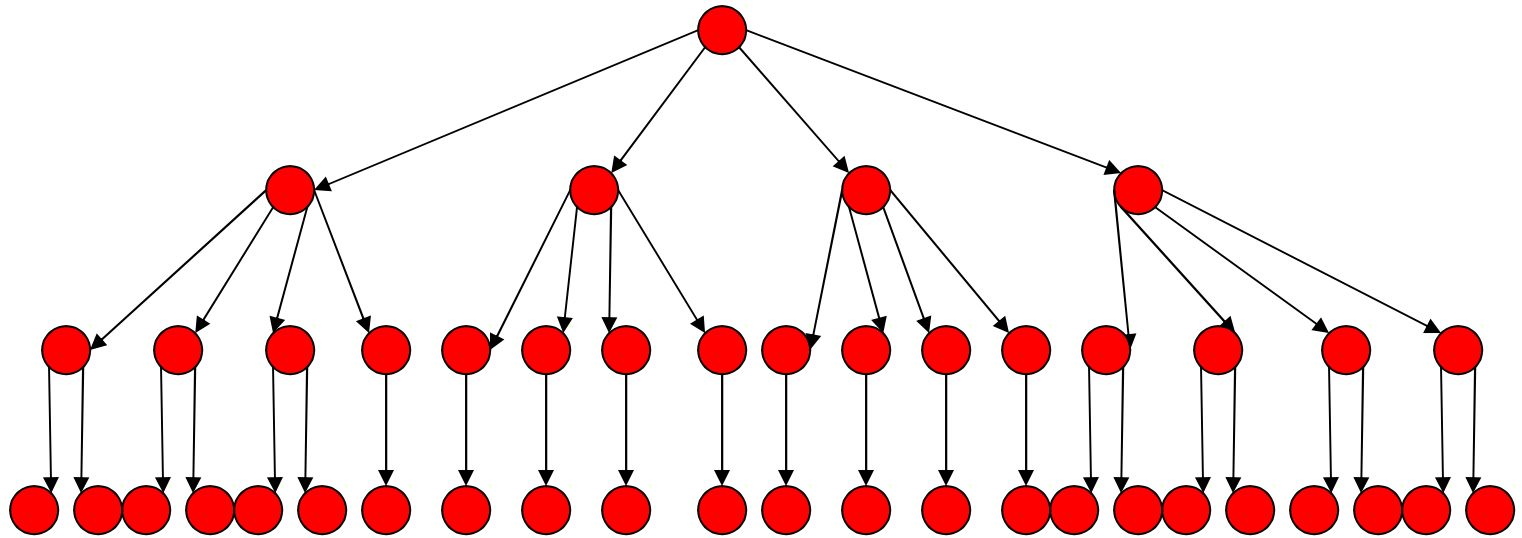
Spidering

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).

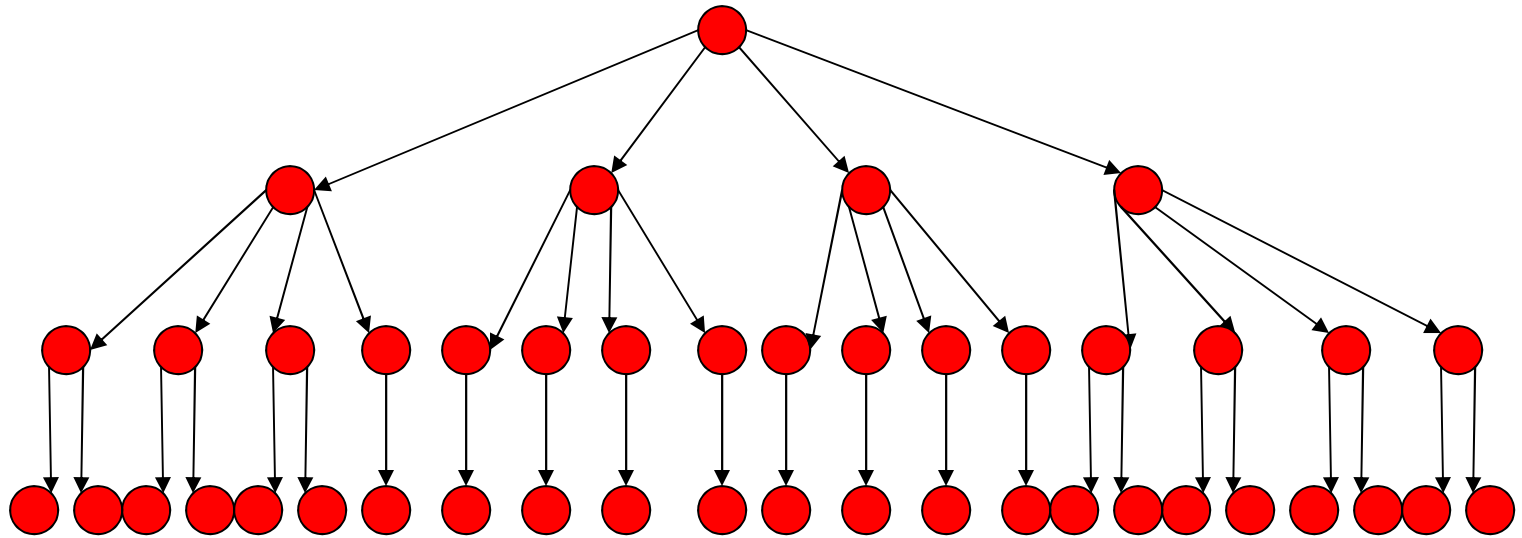
Search Strategies

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.
- Depth-first requires memory of only depth times branching-factor (linear in depth) but gets “lost” pursuing a single thread.
- Both strategies implementable using a queue 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.
 - Canonicalize URL by using “redirected” URL from URLConnection
 - Not detect duplicated or mirrored pages.
- Index page using textual content as a key.
 - Requires first downloading page.

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.

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 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>”>`

Java Spider

- Spidering code in ir.webutils package.
- Generic spider in Spider class.
- Does breadth-first crawl from a start URL and saves copy of each page in a local directory.
- This directory can then be indexed and searched using VSR InvertedIndex.
- Main method parameters:
 - -u <start-URL>
 - -d <save-directory>
 - -c <page-count-limit>

Java Spider (cont.)

- Robot Exclusion can be invoked to prevent crawling restricted sites/pages.
 - -safe
- Specialized classes also restrict search:
 - **SiteSpider**: Restrict to initial URL host.
 - **DirectorySpider**: Restrict to below initial URL directory.

Link Canonicalization

- Canonicalize URL by using “redirected” URL returned by an established Java URLConnection.
- 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>

Link Extraction in Java

- Java Swing contains an HTML parser.
- Parser uses “call-back” methods.
- Pass parser an object that has these methods:
 - `HandleText(char[] text, int position)`
 - `HandleStartTag(HTML.Tag tag, MutableAttributeSet attributes, int position)`
 - `HandleEndTag(HTML.Tag tag, int position)`
 - `HandleSimpleTag (HTML.Tag tag, MutableAttributeSet attributes, int position)`
- When parser encounters a tag or intervening text, it calls the appropriate method of this object.

Link Extraction in Java (cont.)

- In `HandleStartTag`, if it is an “A” tag, take the `HREF` attribute value as an initial URL.
- Complete the URL using the base URL:
 - `new URL(URL baseUrl, String relativeURL)`
 - Fails if `baseUrl` ends in a directory name but this is not indicated by a final “/”
 - Append a “/” to `baseUrl` if it does not end in a file name with an extension (and therefore presumably is a directory).

Cached File with Base URL

- Store copy of page in a local directory for eventual indexing for retrieval.
- BASE tag in the header section of an HTML file changes the base URL for all relative pointers:
 - `<BASE HREF=“<base-URL>”>`
- This is specifically included in HTML for use in documents that were moved from their original location.

Java Spider Trace

- As a simple demo, SiteSpider was used to collect 100 pages starting at: [UT CS Faculty Page](#)
- See trace at:
<http://www.cs.utexas.edu/users/mooney/ir-course/spider-trace.txt>
- A larger crawl from the same page was used to assemble 800 pages that are cached at:
 - </u/mooney/ir-code/corpora/cs-faculty/>

Servlet Web Interface Demo

- Web interface to using VSR to search directories of cached HTML files is at:
 - <http://www.cs.utexas.edu/users/mooney/ir-course/search.html>
- The Java Servlet code supporting this demo is at:
 - `/u/ml/servlets/irs/Search.java`

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.

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.cnn.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.

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.
- Robosurfer assignment in AI course.

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.