Information Retrieval

INFO 4300 / CS 4300

- Last class
  - Search engine architecture...finished.
  - Web crawlers
    - » Retrieving web pages
- Today
  - » Crawling the web
    - Complications
    - Desktop crawlers
    - Document feeds

Web Crawler

- Starts with a set of ***seeds***, which are a set of URLs given to it as parameters
- Seeds are added to a URL ***request queue***
- Crawler starts fetching pages from the request queue
- Downloaded pages are parsed to find link tags that might contain other **useful** URLs to fetch
- New URLs added to the crawler's request queue, or ***frontier***
- Continue until no more new URLs or disk full

Web Crawling

- Web crawlers spend a lot of time waiting for responses to requests
- To reduce this inefficiency, web crawlers use threads and fetch hundreds of pages at once
- Crawlers could potentially flood sites with requests for pages
- To avoid this problem, web crawlers use ***politeness policies***
  - e.g., delay between requests to same web server

Controlling Crawling

- Even crawling a site slowly will anger some web server administrators, who object to any copying of their data
- robots.txt file can be used to control crawlers

robots.txt

- Protocol for giving crawlers/spiders ("robots") limited access to a website, originally from 1994
  - www.robotstxt.org/wc/norobots.html
- Website announces its request for what can(not) be crawled
  - For a server, create a file `/robots.txt`
  - This file specifies access restrictions

robots.txt - example

```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public/

User-agent: FavoredCrawler
Disallow:

Sitemap: http://mysite.com/sitemap.xml.gz
```

www.robotstxt.org

Simple Crawler Thread

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

Information Retrieval

INFO 4300 / CS 4300

- Last class
  - Search engine architecture...finished.
  - Web crawlers
    - » Retrieving web pages
- Today
  - » Crawling the web
    - ◆ Complications
    - ◆ Desktop crawlers
    - ◆ Document feeds

Complications

- Freshness
- Focused crawling
- Deep web
- Distributed crawling

Freshness

- Web pages are constantly being added, deleted, and modified
- Web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the ***freshness*** of the document collection
  - *stale* copies no longer reflect the real contents of the web pages

Freshness

- HTTP protocol has a special request type called HEAD that makes it easy to check for page changes
  - returns information about page, not page itself

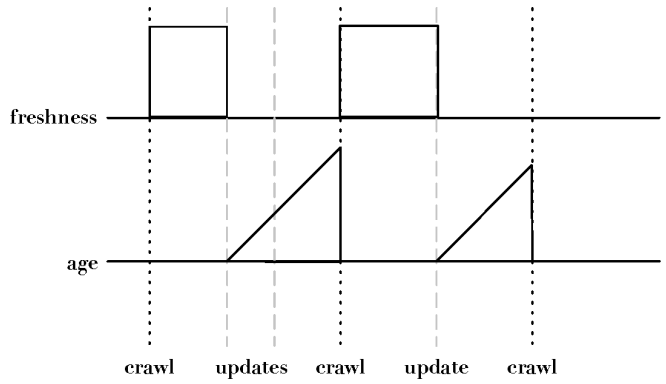
Client request:
```
HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
```

Server response:
```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Freshness

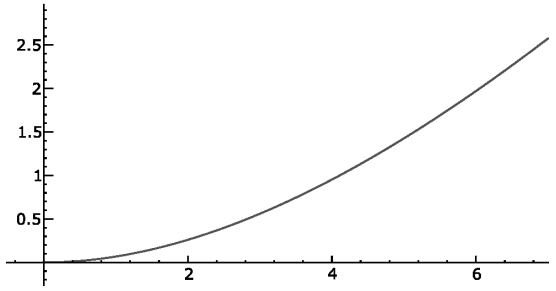
- Not possible to constantly check all pages
  - must check important pages and pages that change frequently
- **Freshness** metric: the proportion of pages that are fresh, i.e., up-to-date
- Optimizing for this metric can lead to bad decisions, such as not crawling popular sites
- **Age** is a better metric

Freshness vs. Age



Age

- Older a page gets, the more it costs not to crawl it
  - e.g., expected age with mean change frequency $\lambda = 1/7$ (one change per week)



Focused Crawling

- Attempts to download only those pages that are about a particular topic
  - used by *vertical search* applications
- Rely on the fact that pages about a topic tend to have links to other pages on the same topic
  - popular pages for a topic are typically used as seeds
- Crawler uses ***text classifier*** to decide whether a page is on topic

Deep Web

- Sites that are difficult for a crawler to find are collectively referred to as the ***deep*** (or ***hidden***) *Web*
  - much larger than conventional Web
- Three broad categories:
  - private sites
    - » no incoming links, or may require log in with a valid account
  - form results
    - » sites that can be reached only after entering some data into a form
  - scripted pages
    - » pages that use JavaScript, Flash, or another client-side language to generate links

Sitemaps

- Sitemaps contain lists of URLs and data about those URLs, such as modification time and modification frequency
- Generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check a page for changes

Sitemap Example

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/items?item=truck</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/items?item=bicycle</loc>
    <changefreq>daily</changefreq>
  </url>
</urlset>
```