Crawling the Web

Information Retrieval

Universal Resource Identifiers

- Universal Resource Identifier (URI)
  - DEF: A string of characters used to identify a resource
- Examples of URIs:
  - http://www.ics.uci.edu (URL)
  - ISBN 0-486-27777-3 (URN)
  - ftp://ftp.ics.uci.edu (URL)
- URL (locator) vs URN (name)
  - Locator must specify *where* the resource is
- We are going to focus on URLs
  - But “URI” might slip in as synonym

Anatomy of a URL

- Syntax:
  - scheme://domain:port/path?query_string#fragment_id

    authority
  - (slightly more complicated than this)
- Full spec:
  - http://www.w3.org/Addressing/URL/url-spec.txt

Anatomy of a URL

- http://www.ics.uci.edu/~lopes

on a web server

no port! just domain

path

query

- http://calendar.ics.uci.edu/calendar.php?type=month&calendar=1&category=&month=02&year=2013

- Domains and subdomains:
  - calendar.ics.**uci.edu**

Domain name

Different Flavors of Web Data Collection

- Data dumps
- URL downloads
- Web APIs
- Web Crawling

Data dumps

- Sites may package their data periodically and provide it as a “dump”
  - Example: Wikipedia

URL Downloads

- Two step process:
  1. Crawl to find out the URLs of specific resources
  2. Run a downloader that takes that list and downloads the resources
- Example: "crawling" sourceforge for source code
- Some sites use regular URLs. E.g. Google Code
  - http://code.google.com/p/**crawler4j**/downloads/list
  - http://code.google.com/p/**python-for-android**/downloads/list
  - …
- Doesn't need to be source code; can be papers, pages, etc.
  - http://link.springer.com/chapter/10.1007/978-3-642-34213-4_1
  - http://link.springer.com/chapter/10.1007/978-3-642-34213-4_2
  - …

Web APIs

- Sites may provide REST interfaces for getting at their data
  - Usually higher-level: avoids having to parse HTML
  - Usually restrictive: only part of the data
- Examples:
  - Facebook Graph API
    - My data in facebook api
    - More examples
  - Youtube API
  - Twitter API
  - …

Web Crawling

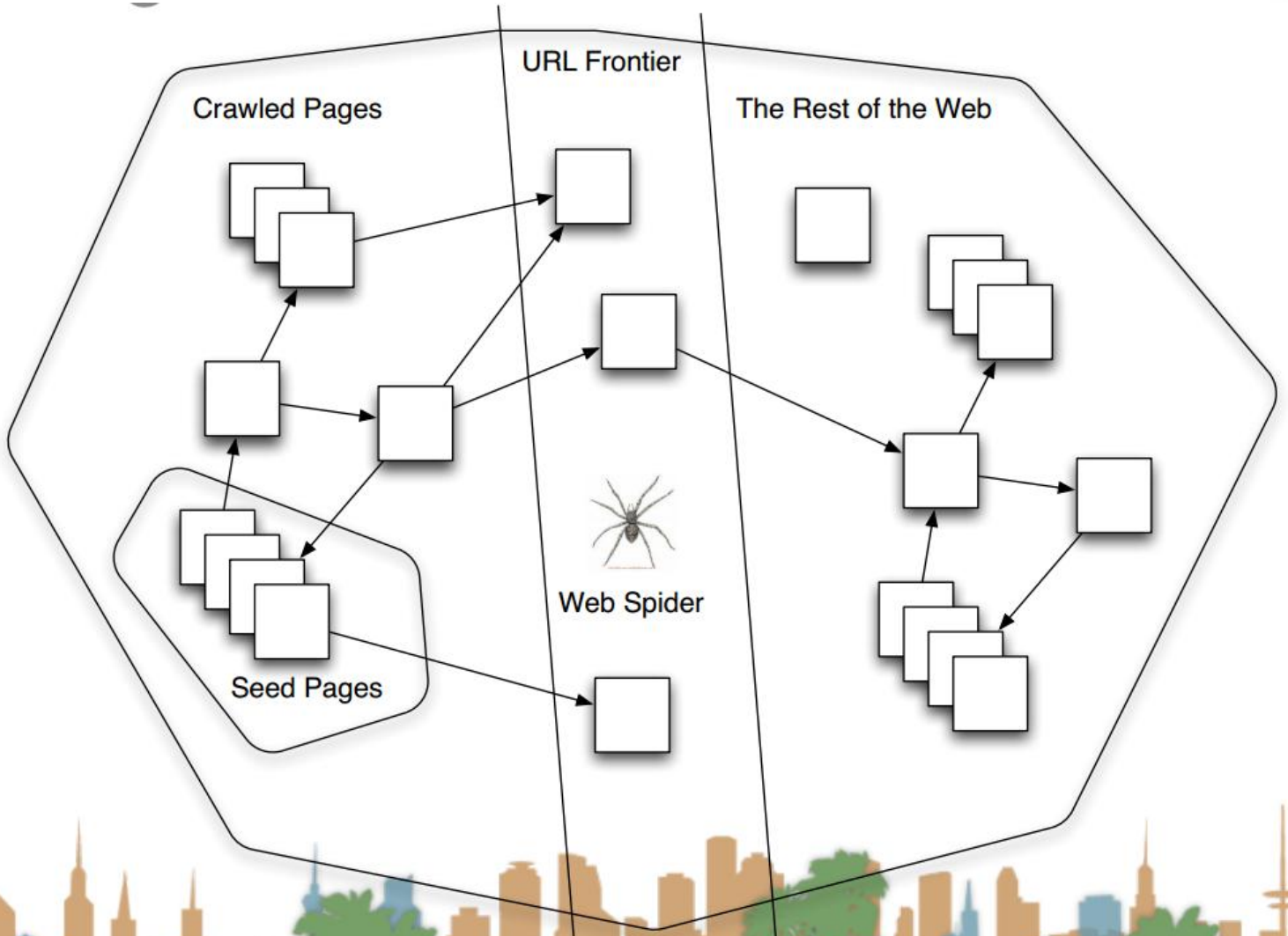
- Like people, getting HTML pages and other documents and discovering new URLs as it goes
  - Good for changing collections
  - Good for unknown documents
- Web admins don’t like crawlers
  - Crawlers consume resources that are meant for people
  - More on this...

Basic Crawl Algorithm

- Initialize a queue of URLs (seeds)
- Repeat until no more URLs in queue:
  - Get one URL from the queue
  - If the page can be crawled, fetch associated page
  - Store representation of page
  - Extract URLs from page and add them to the queue

- Queue = “frontier”

Basic Crawl Algorithm



Pseudo Code

```
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
```

Pseudo Code

```
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
```

Permission to crawl

- Robots Exclusion Standard aka **robots.txt**
  - Sites may have that file at the root. Examples:
    - http://www.cnn.com/robots.txt
    - http://en.wikipedia.org/robots.txt
  - Very simple syntax:
    - http://www.robotstxt.org/robotstxt.html
  - Honor basis!
    - It's not a security mechanism

Information to crawlers

- Sitemaps (introduced by Google)
- Also listed in robots.txt
- Allow web masters to send info to crawlers
  - Location of pages that might not be linked
  - Relative importance
  - Update frequency
- Example:
  - http://www.cnn.com/robots.txt

Basic algorithm is…

- Theoretically correct
- **Seriously lacking to use in practice**
  1. Will upset web admins (impolite)
     - It's abusing the web servers
  2. Very slow
     - 1 page at a time
  3. Will get caught in traps and infinite sequences
  4. Will fetch duplicates without noticing
  5. Will bring in data noise
  6. Will miss content due to client-side scripting

1. Politeness

- Avoid hitting any site too often
  - Sites are for people, not for bots
- Ignore politeness ➔ Denial of service (DOS) attack
- Be polite ➔ Use artificial delays

2. Performance (I)

- Back of the envelope calculation:
  - 1 page fetch = 500ms
  - How much time to crawl 1 million pages?
    - (it's worse than that... Unresponsive servers)
- Most of the time, the crawler thread is waiting for the network data
- Solution: multi-threaded or distributed crawling
  - Politeness harder control

2. Performance (II)

- Domain Name lookups
  - Given a domain name, retrieve its IP address
    - www.ics.uci.edu -> 128.195.1.83
- Distributed set of servers
  - Latency can be high (2 secs is not unusual)
- Common implementations are blocking
  - One request at a time
  - Result is cached
- Back of the envelope calculation:
  - 1 DNS lookup ➔ 800ms
  - How much time to lookup the entire Web?

3. Crawler traps

- Traps the crawler on the site forever
  - Web server responds with ever changing URLs and content
  - May be intentional or unintentional
    - E.g. the ICS calendar is a crawler trap

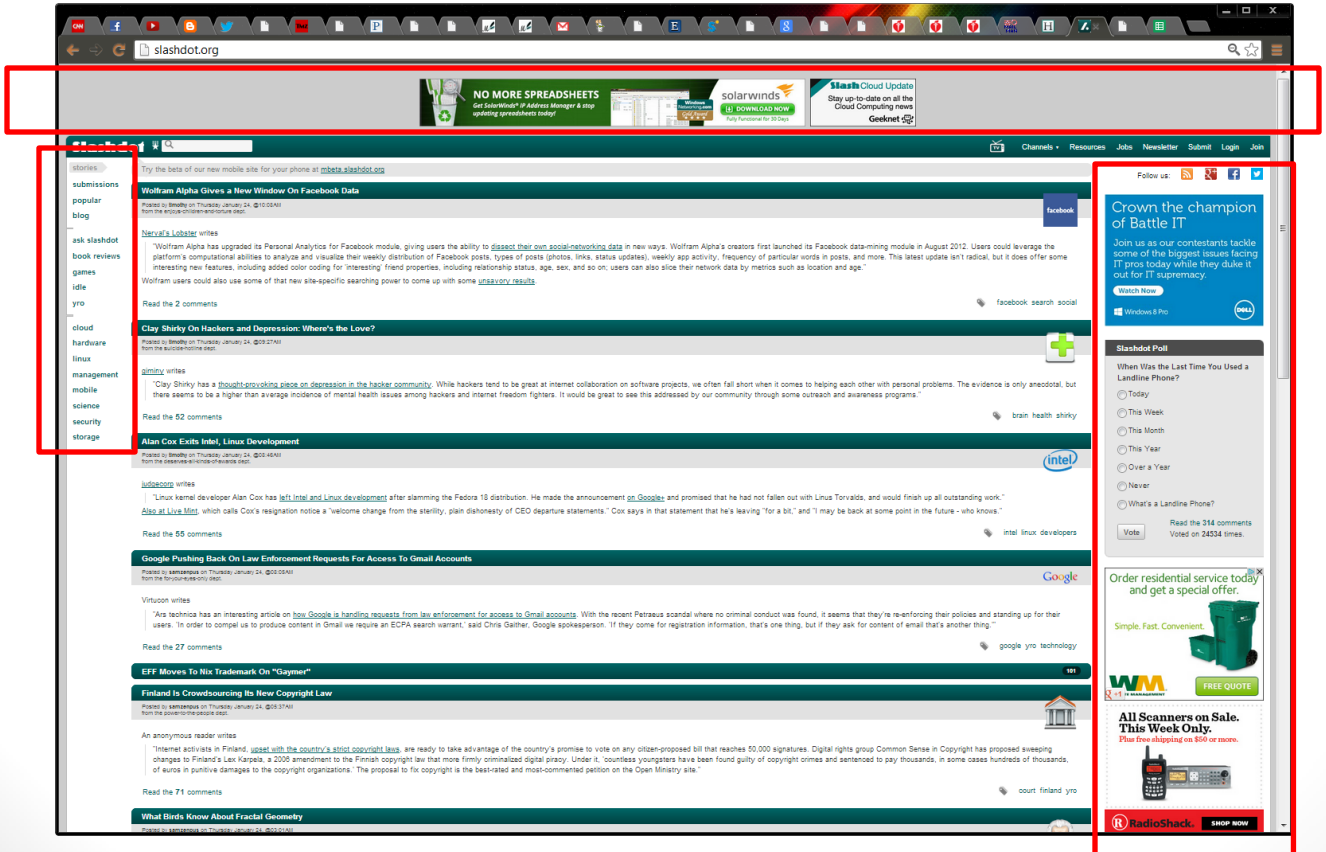
- See http://www.fleiner.com/bots/

4. Duplicate Detection

- Duplication and near-duplication is widespread
  - Copies, mirror sites, versions, spam, plagiarism...
  - Studies: 30% of Web pages are [near-]duplicates of the other 70%
  - Little or no value, noise
- Detection
  - Detection of exact duplication is easy, but exact duplication is rare
    - Hashes, checksums
  - Detection of near-duplicates is hard
    - Page fingerprints

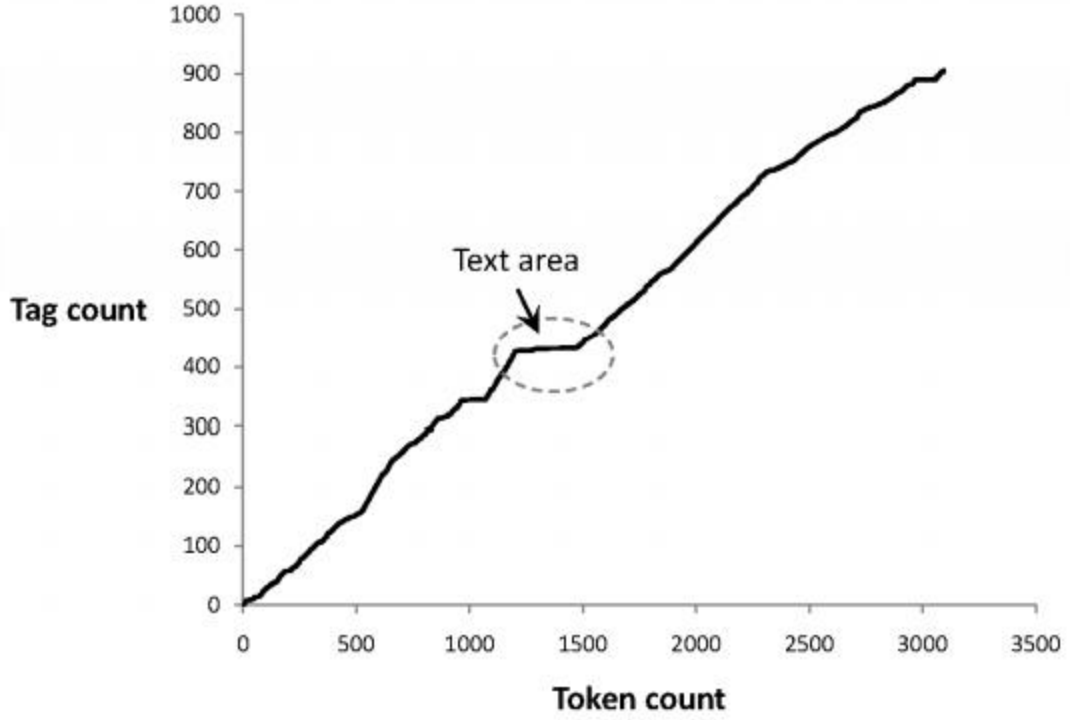
5. Data Noise

- Web pages have content not directly related to the page
  - Ads, templates, etc
  - Noise negatively impacts information retrieval



Finding Content Blocks

- Technique 1: Cumulative distribution of tags



- Other techniques in literature

6. Client-Side Scripting

- Modern web sites are heavily scripted (JavaScript)
  - Content behind XMLHttpRequests
- To get to that content crawlers must run the scripts
  - Hard thing to do (user interaction)
  - Crawler4j doesn’t do it

The Deep Web

- Places where crawlers rarely go...
  - Content behind login forms
  - Content behind JavaScript
  - Sites that aren’t linked from anywhere
- It is estimated that the deep web is larger than the shallow web