Information Retrieval and Web Search

Web Crawling

Instructor: Rada Mihalcea

(some of these slides were adapted from Ray Mooney’s IR course at UT Austin)
The Web by the Numbers

- Web servers – 634 million
- Users worldwide – 2.4 billion
- Size of the Web – 5,000,000 TB (geekwire.com)
  - About 20% text
  - 7,500 TB in deep Web
- Search – 1.2 trillion searches on Google in 2012

http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/
The Web by the Numbers

• Emails
  – Total emails per day 154 billion (radicati.com)
  – Percentage spam 64.8%

• Social media:
  – Active users on Facebook
    • 1 billion
  – Active users on Twitter
    • 200 million

• Number of smartphone subscribers
  – 1.1 billion

http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/
Top Ten Languages on the Web

- English: 27%
- Chinese: 23%
- Korean: 2%
- Other: 18%
- German: 4%
- French: 3%
- Arabic: 3%
- Portugese: 4%
- Russian: 3%
- Japanese: 5%
- Spanish: 8%

Source: internetworldstats.com ~ updated May 31, 2011
Language Growth over the Last Decade

- English: 301%
- Chinese: 1479%
- Spanish: 807%
- Japanese: 110.70%
- Portuguese: 990%
- German: 174%
- Russian: 1826%
- Arabic: 2501%
- French: 398%
- Rest of the Languages: 589%

internetworldstats.com ~ updated May 31, 2011
Web Challenges for IR

• **Distributed Data**: Documents spread over millions of different web servers.

• **Volatile Data**: Many documents change or disappear rapidly (e.g. dead links).

• **Large Volume**: Billions of separate documents.

• **Unstructured and Redundant Data**: No uniform structure, HTML errors, up to 30% (near) duplicate documents.

• **Quality of Data**: No editorial control, false information, poor quality writing, typos, etc.

• **Heterogeneous Data**: Multiple media types (images, video, VRML), languages, character sets, etc.
Web Search Using IR

The crawler represents the main difference compared to traditional IR.
Crawlers (Robots/Bots/Spiders)

- Start with a comprehensive set of root URLs
- Follow all links on these pages recursively to find additional pages.
- Index/Process 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).
What any Crawler *Must* Do

- Be **Polite**: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect *robots.txt* (more on this shortly)

- Be **Robust**: Be immune to spider traps and other malicious behavior from web servers
What any Crawler Should Do

- Be capable of **distributed** operation: designed to run on multiple distributed machines
- Be **scalable**: designed to increase the crawl rate by adding more machines
- **Performance/efficiency**: permit full use of available processing and network resources
What any Crawler Should Do

• Fetch pages of “higher quality” first
• Continuous operation: Continue fetching fresh copies of a previously fetched page
• Extensible: Adapt to new data formats, protocols
Crawling Picture

Unseen Web

Seed Pages

URLs crawled and parsed

Crawling thread

URL frontier
Processing Steps in Crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (URLs)
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)

E.g., only crawl .edu, obey robots.txt, etc.
Basic Crawl Architecture

WWW

DNS

Fetch

Parse

Doc FP’s

robots filters

URL set

Content seen?

URL filter

Dup URL elim

URL Frontier

Sec. 20.2.1 13
DNS (Domain Name Server)

- A lookup service on the internet
  - Given a URL, retrieve its IP address
  - Service provided by a distributed set of servers – thus, lookup latencies can be high (even seconds)

- Common OS implementations of DNS lookup are blocking: only one outstanding request at a time
  - Biggest bottleneck in Web crawling

- Solutions
  - DNS caching
  - Batch DNS resolver – collects requests and sends them out together
Traversing Strategies

- The Web is a graph
  - Graph traversal strategies

- **Breadth-first** explores uniformly outward from the root page but requires memory of all nodes on the previous level (exponential in depth). Standard crawling method.

- **Depth-first** requires memory of only depth times branching-factor (linear in depth) but gets “lost” pursuing a single thread.

- Both strategies can be easily implemented using a queue of links (URLs).
Traversa l Algorithm

Initialize queue (Q) with initial set of known URLs.
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 are 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.
Directed/Focused Crawling

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

- 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.
  - Related to Topic Tracking and Detection
Link-Directed Crawling

- Monitor links and keep track of in-degree and out-degree of each page encountered.
- Sort queue to prefer popular pages with many incoming links (authorities).
- Sort queue to prefer summary pages with many outgoing links (hubs).
- Related to Google’s PageRank algorithm
Parsing and Link Extraction

• Must find all links in a page and extract URLs.
  – <a href="http://web.eecs.umich.edu/~mihalcea/courses/498IR”>
  – <frame src="site-index.html”>

• Must complete relative URLs using current page URL:
  – <a href="proj3”> to http://web.eecs.umich.edu/~mihalcea/courses/498IR/proj3
  – <a href="../cs5343/syllabus.html”> to http://web.eecs.umich.edu/~mihalcea/courses/cs5343/syllabus.html
URL Syntax

- A URL has the following syntax:
  - `<scheme>://<authority><path>?<query>#<fragment>`

- 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://web.eecs.umich.edu/~mihalcea/courses/498IR
  - http://web.eecs.umich.edu/~mihalcea/courses/498IR/

- Internal page fragments (references) removed:
  - http://web.eecs.umich.edu/~mihalcea/welcome.html#coursess
  - http://web.eecs.umich.edu/~mihalcea/welcome.html
Anchor Text Indexing

• Extract anchor text (between <a> and </a>) of each link followed.

• Increases content more for popular pages with many in-coming links, increasing recall of these pages.

• Anchor text is usually descriptive of the document to which it points.
  – Sometimes anchor text is not useful: e.g., “click here”

• Add anchor text to the content of the destination page to provide additional relevant keyword indices.

• Used by Google:
  – <a href="http://www.microsoft.com">Evil Empire</a>
  – <a href="http://www.ibm.com">IBM</a>
Restricting Crawling

• You can restrict crawler to a particular site.
  – Remove links to other sites from Q.

• You can restrict crawler to a particular directory.
  – Remove links not in the specified directory.

• Obey page-owner restrictions (robot exclusion).
Crawler Politeness

• Explicit politeness: specifications from webmasters on what portions of site can be crawled
  – robots.txt

• Implicit politeness: even with no specification, avoid hitting any site too often
Robot Exclusion

- Web sites and pages can specify that robots should not crawl/index certain areas.

- Two components:
  - **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.ebay.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:
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

- 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.
Avoiding Page Duplication

- The web is full of duplicated content
- Must detect when revisiting a page that has already been crawled (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 normalize URLs (e.g. delete ending “/”)
- Index page using textual content as a key.
  - Requires first downloading page.
Near-Duplicates

• There are also many cases of near duplicates
  – E.g., Last modified date the only difference between two copies of a page

• *Near-Duplication*: Approximate match

• Compute syntactic similarity with an edit-distance measure

• Use similarity threshold to detect near-duplicates
  • E.g., Similarity > 90% => Documents are “near duplicates”
Computing Document Similarity

• Features:
  - Segments of a document (natural or artificial breakpoints)
  - Shingles (Word N-Grams)
  - \(\textit{a rose is a rose is a rose} \rightarrow\) 4-grams are
    - a\_rose\_is\_a
    - rose\_is\_a\_rose
    - is\_a\_rose\_is
    - a\_rose\_is\_a

• Similarity Measure between two docs (= two sets of shingles)
  - Set intersection
  - Specifically \((\text{Size\_of\_Intersection} / \text{Size\_of\_Union})\) (Jaccard)
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 URLs to threads to guarantee equitable distribution of requests across different hosts to maximize throughput 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.
Keeping Crawled Pages Up to Date

- Web is very dynamic: many new pages, updated pages, deleted pages, etc.

- Periodically check crawled 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.