Introduction to Big Data

Pooya Jamshidi

pooya.jamshidi@ut.ac.ir

Ilam University

School of Engineering,
Computer Group

March 2, 2025

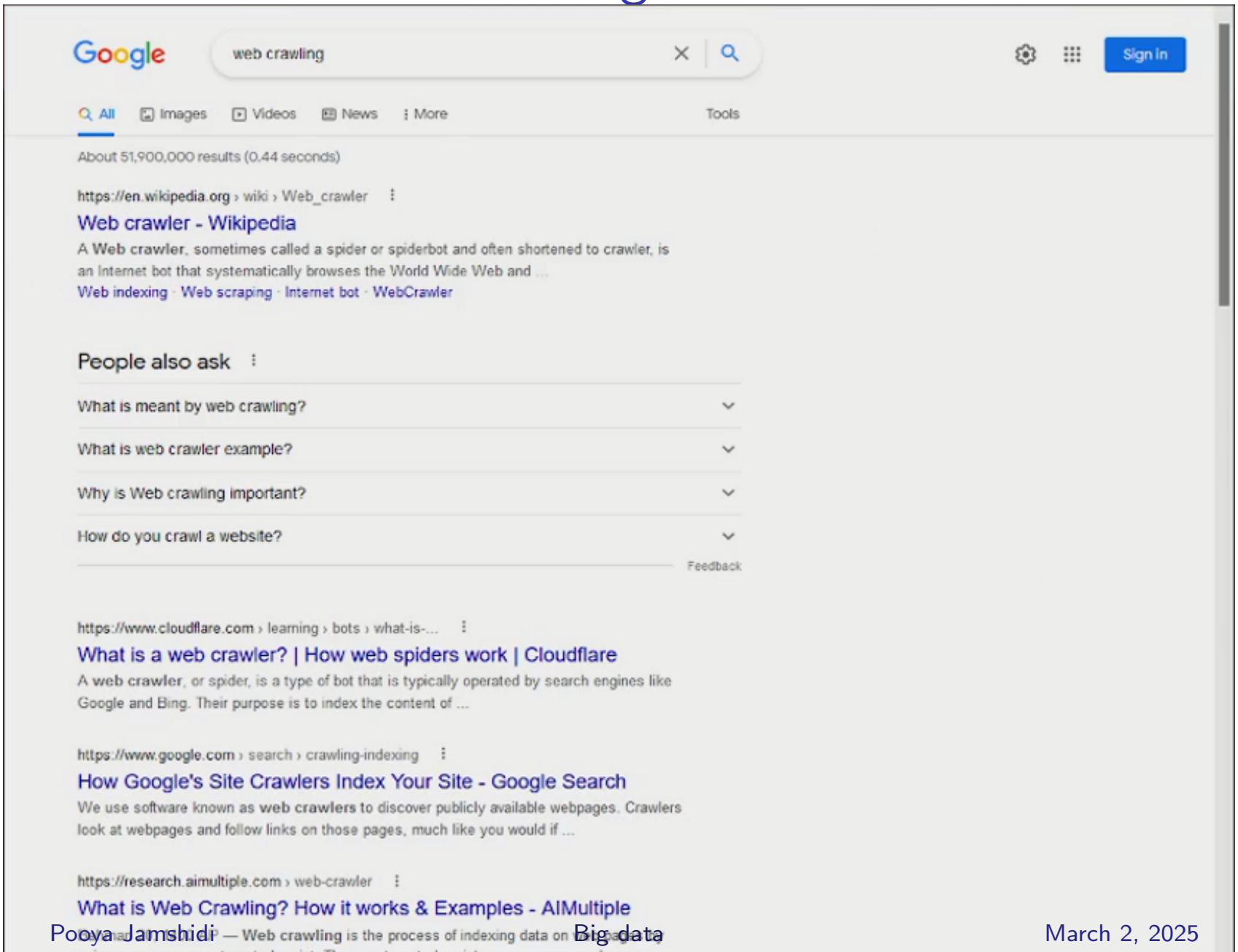
Web Crawling

Outline

- Motivation
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers
- Crawling in action (wget, curl, httrack, Nutch, Heritrix, and Stormcrawler)

- **Motivation**
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers
- Crawling in action (wget, curl, httrack, Nutch, Heritrix, and Stormcrawler)

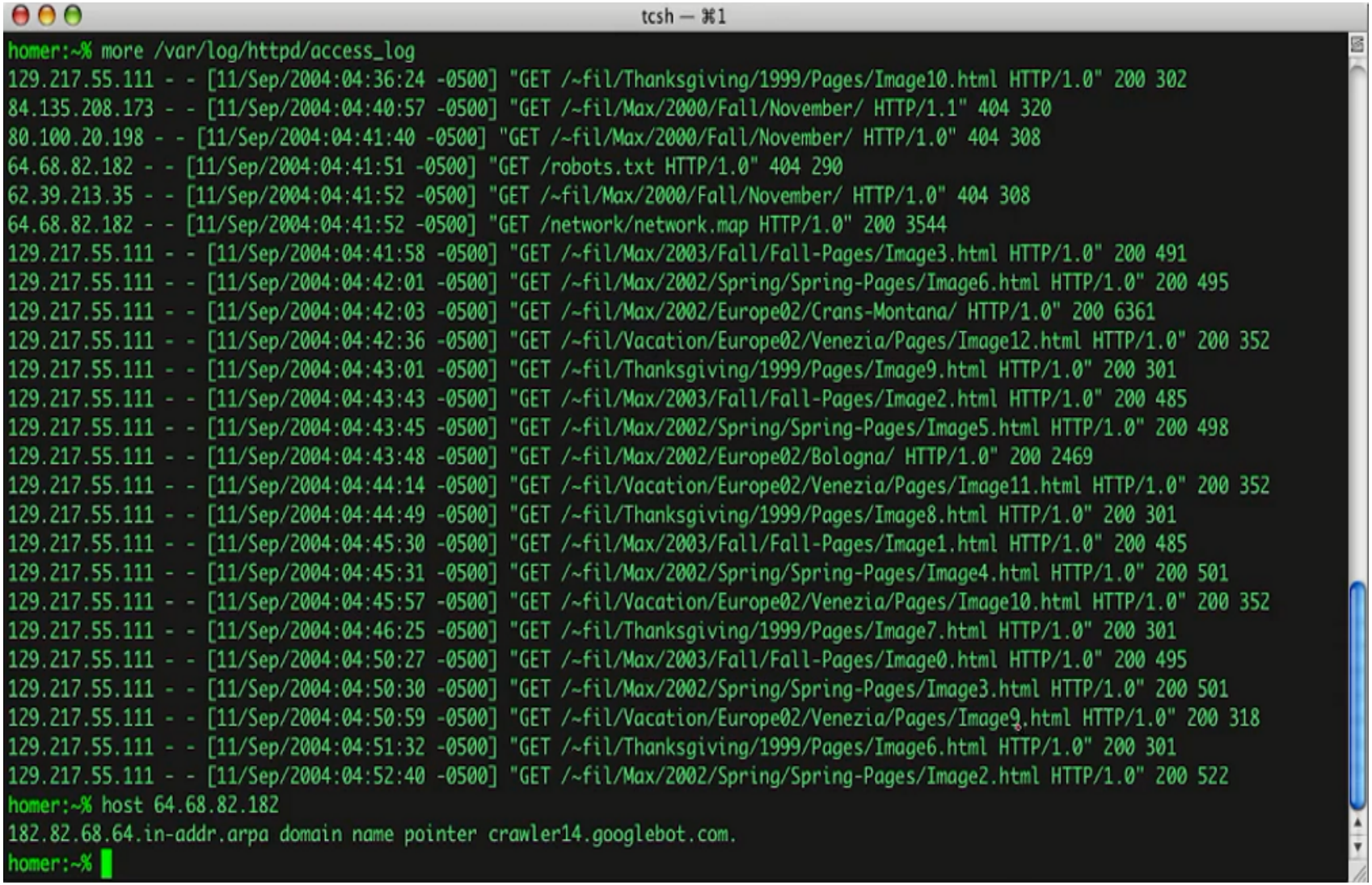
Some Google Serach



Many Names

- Crawler
- Spider
- Robot (or bot)
- Web agent
- Wanderer, worm, ...
- And famous instances:
  - Googlebot
  - Bingbot
  - DuckDuckBot
  - Slurp (Yahoo!)
  - Baiduspider
  - YandexBot
  - facebookexternalhit(Why?)

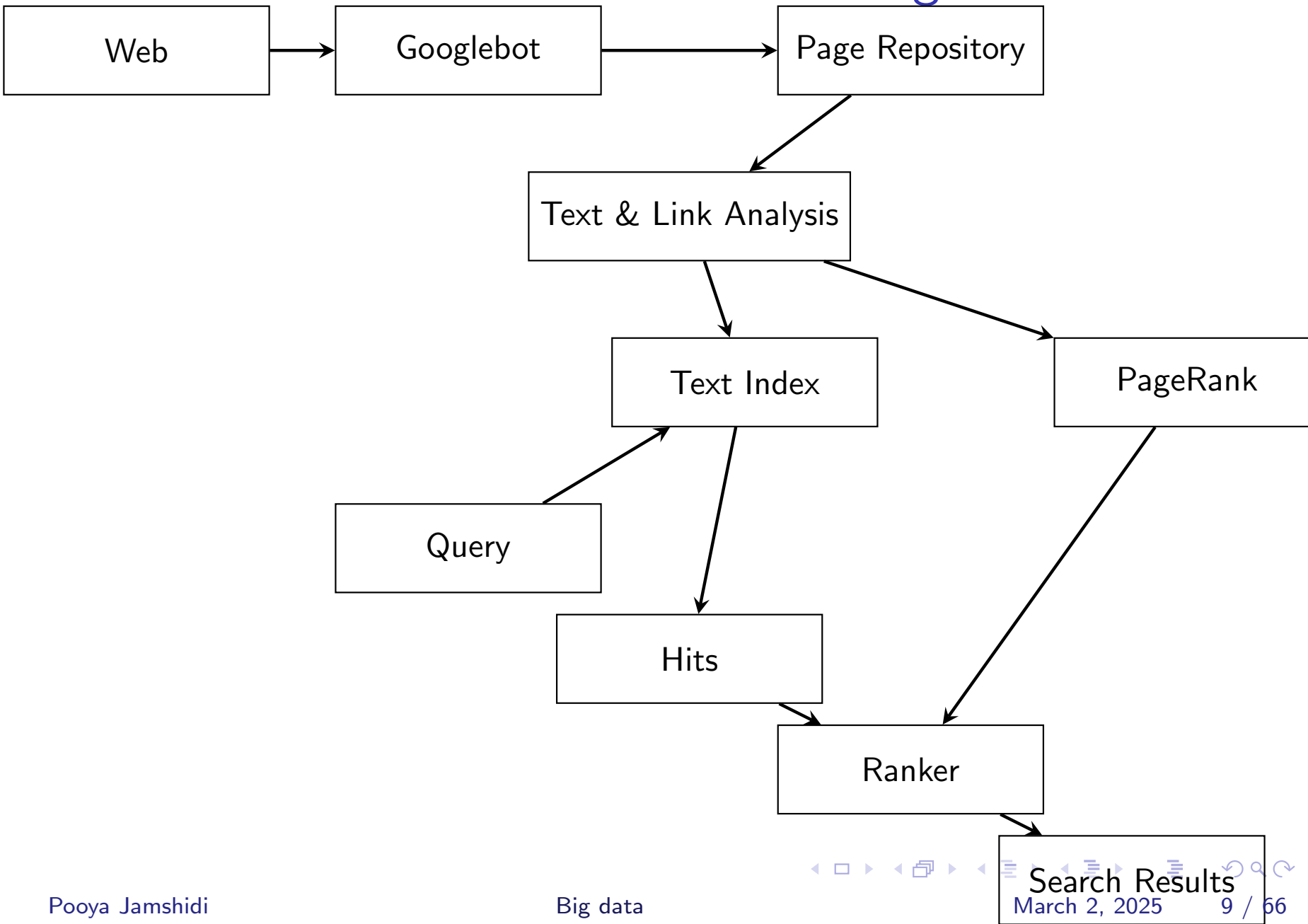
Googlebot & You



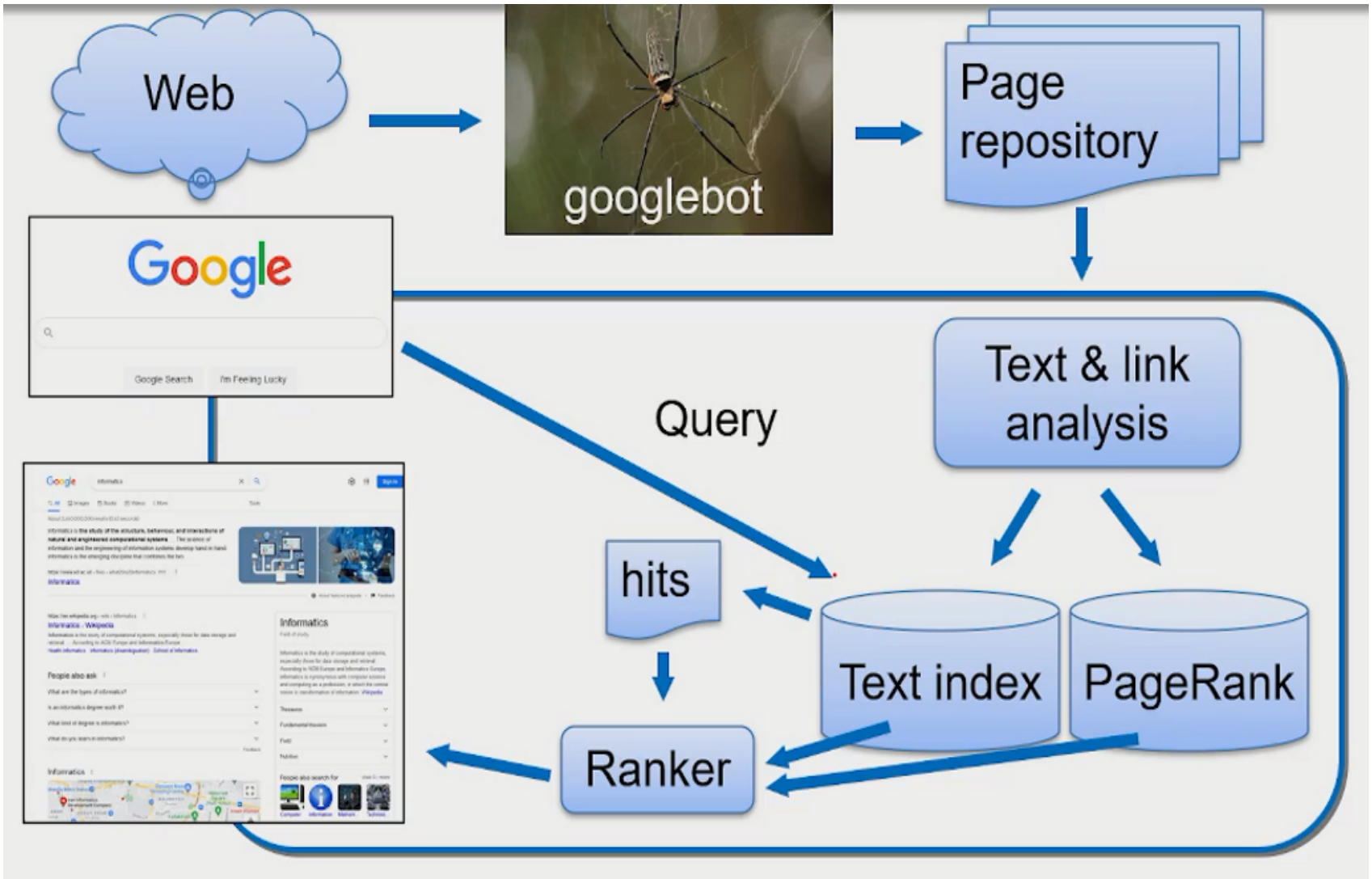
Motivation for Crawlers

- Support universal search engines (Google, Bing, Yandex, Baidu, Ask, etc.)
- Vertical specialized search engines, e.g., news, shopping, papers, recipes, reviews, etc.
- Business intelligence: Keep track of potential competitors, partners
- Monitor websites of interest
- Evil: Harvest emails for spamming, phishing
- . . . Can you think of some others?

A Crawler Withnin a Search Engine

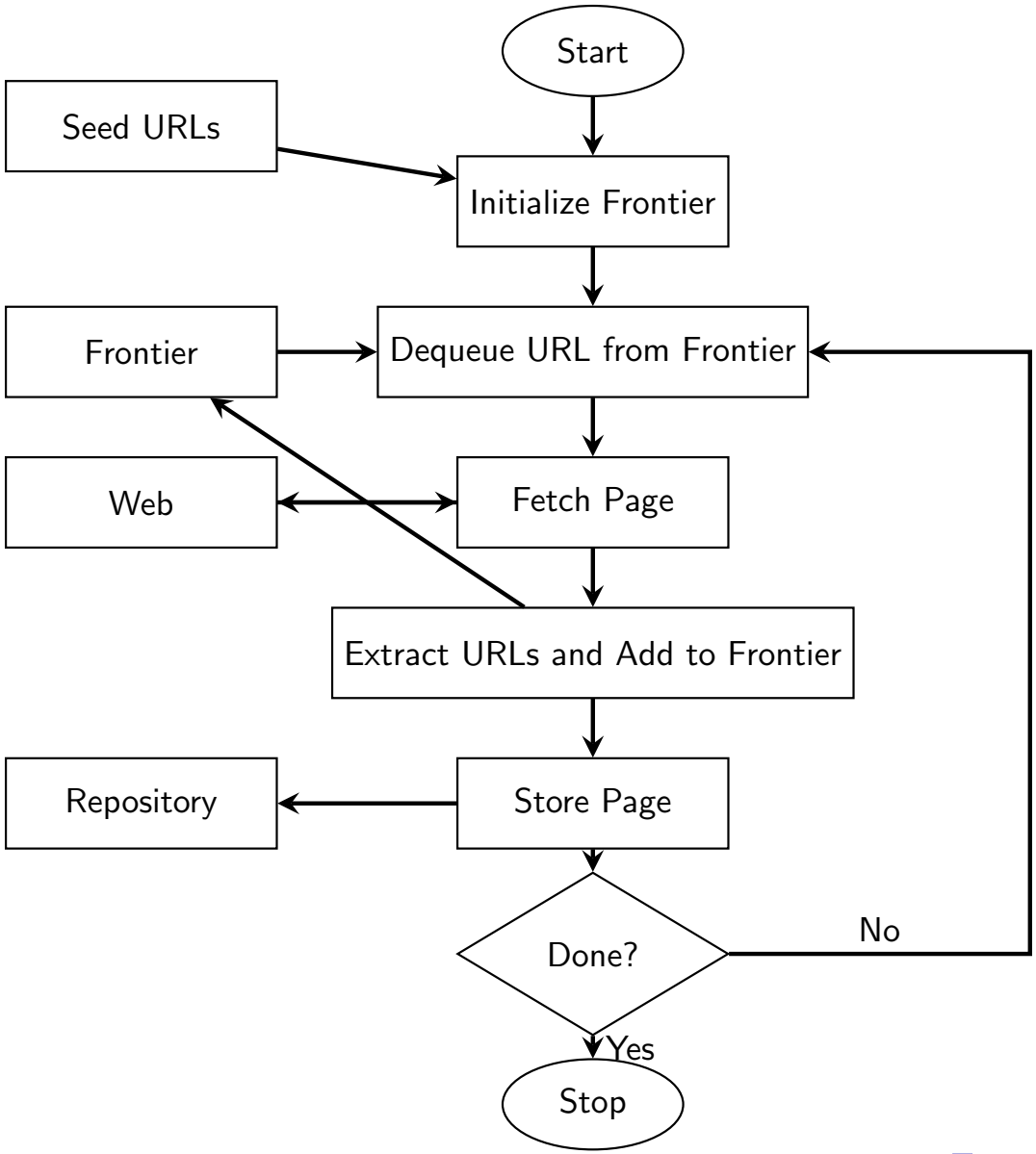


A Crawler Withnin a Search Engine(Cont.)



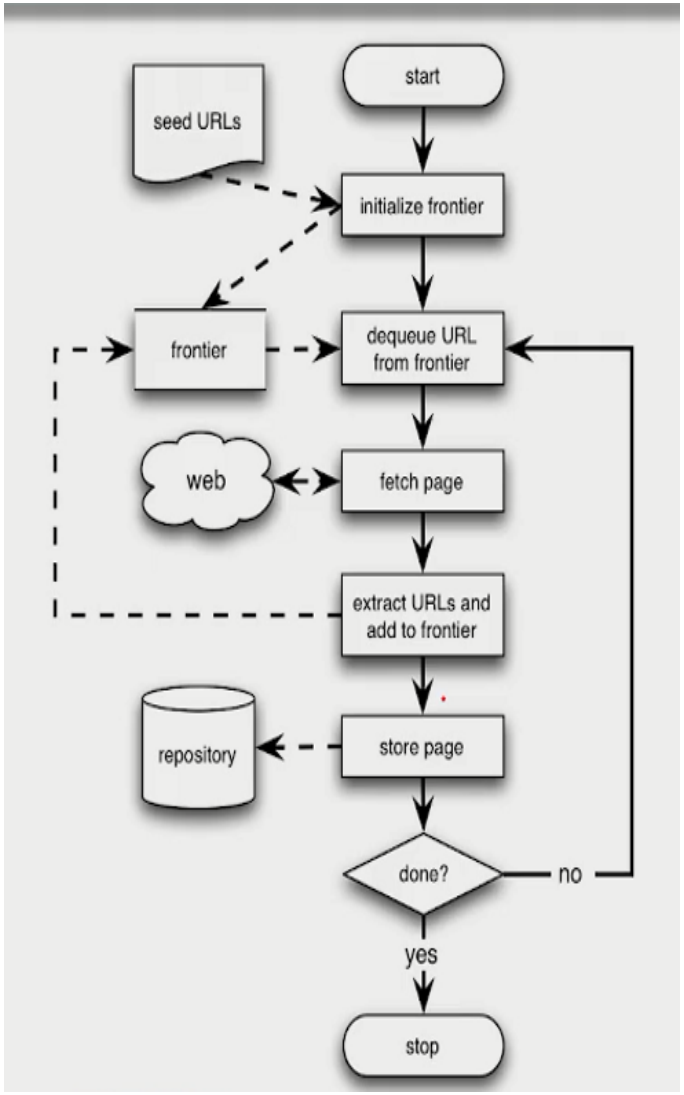
- Motivation
- **Basic crawlers and implementation issues**
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers
- Crawling in action (wget, curl, httrack, Nutch, Heritrix, and Stormcrawler)

Basic Crawlers

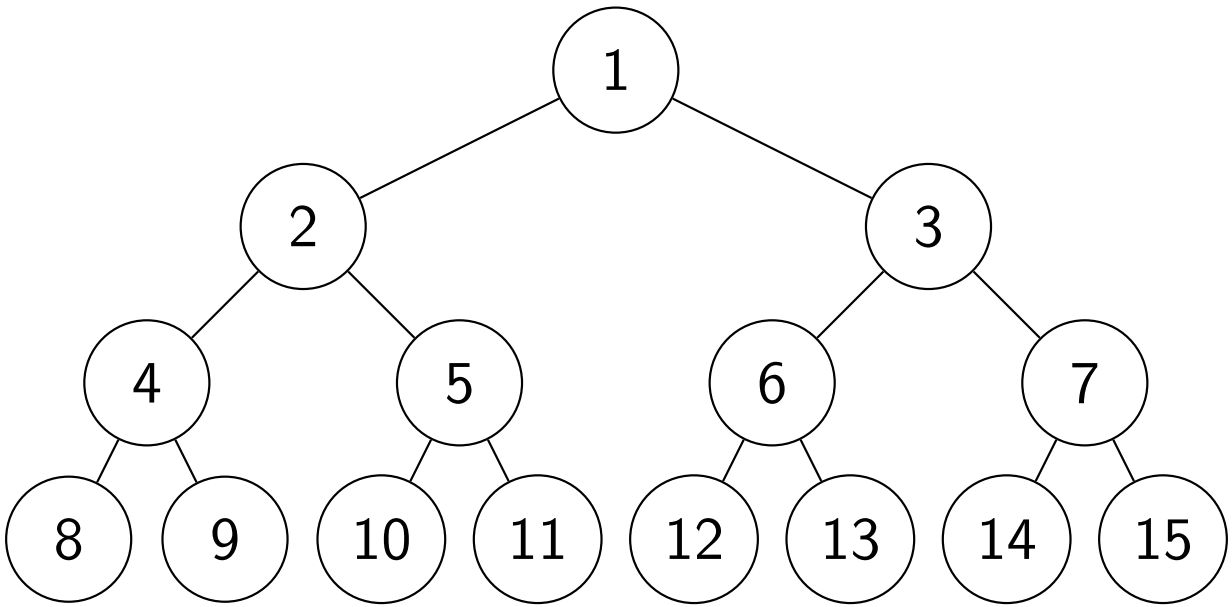


Basic Crawlers

- This is a sequential crawler.
- Seeds can be any list of starting URLs.
- Order of page visits is determined by frontier data structure.
- Stop criterion can be anything.



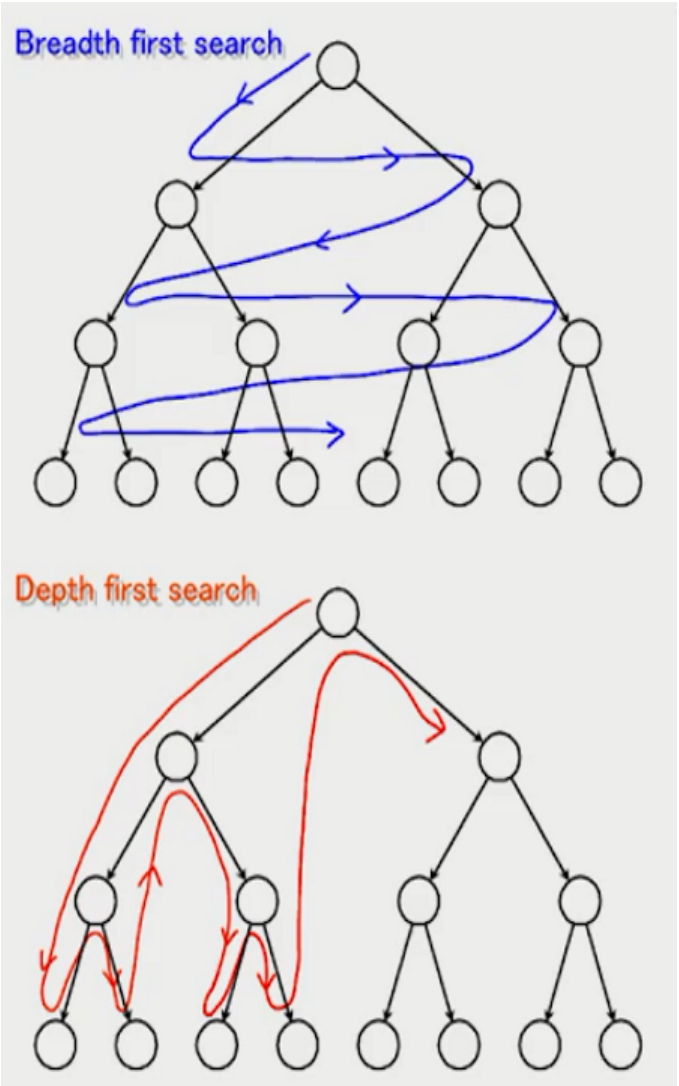
Grph Traversal(BFS or DFS)



- BFS : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
- DFS: 8, 9, 4, 10, 11, 5, 2, 12, 13, 6, 14, 15, 7, 3, 1

Graph Traversal (BFS or DFS?)

- **Breadth-First Search**
  - Implemented with *Queue* (FIFO).
  - Finds pages along shortest paths.
  - If we start with "good" pages, this keeps us close; maybe other good stuff...
- **Depth-First Search**
  - Implemented with *Stack* (LIFO).
  - Wander away ("lost in cyberspace").



Implementation Issues

- Don't want to fetch the same page twice!
  - Keep a lookup table (hash) of visited pages.
  - What if not visited but in the frontier already?
- The frontier grows very fast!
  - May need to prioritize for large crawls.
- Fetcher must be robust!
  - Don't crash if download fails.
  - Timeout mechanism.
- Determine file type to skip unwanted files.
  - Can try using extensions, but not reliable.
  - Can issue `HEAD` HTTP commands to get Content-Type (media type or MIME type) headers, but overhead of extra Internet requests.

Media Type or MIME Type

- A media type (formerly known as MIME type) is a two-part identifier for file formats and format contents transmitted on the Internet.
- **Format**: Consists of a type and a subtype, which is further structured into a tree. A media type can optionally define a suffix and parameters:

```
type "/" [tree "."] subtype ["+" suffix]* [";"
                        parameter]
```

- **Registered types**: `application`, `audio`, `image`, `message`, `multipart`, `text`, `video`, `font`, `example`, `model`.
- **Examples**:
  - `text/html; charset=UTF-8`
  - `application/msword (.doc)`
  - `application/vnd.ms-powerpoint (.ppt)`
  - `audio/mpeg`

Media Type or MIME Type (cont'd)

- **How to get the media type for a remote file?**
  - `$ curl -s -I www.google.com | grep -i "Ĉontent-Type:"`
  - `Content-Type: text/html; charset=ISO-8859-1`
- **How to get the media type for a local file?**
  - File extension
  - **Magic number**: A numeric or string constant that indicates the file type. This number is in the first 512 bytes of the file.
    - **PDF files** start with `"%PDF"` (hex `25 50 44 46`).
    - **PNG files** begin with an 8-byte signature and allow detection of common file transfer problems: `N G \r \n \032 \n` (hex `89 50 4E 47 0D 0A 1A 0A`).
    - Contains various newline characters to permit detecting unwarranted automated newline conversions, such as transferring the file using FTP with the ASCII transfer mode instead of the binary mode.

Media Type or MIME Type(practical Examples)

Linux Way

Terminal Output

```
$ mv test.png test && file test
test:  PNG image data, 64 x 64, 8-bit/color
RGBA, non-interlaced

$ file -i test
test:  image/png; charset=binary
```

Media Type or MIME Type(practical Examples)

Python Way

Python Code

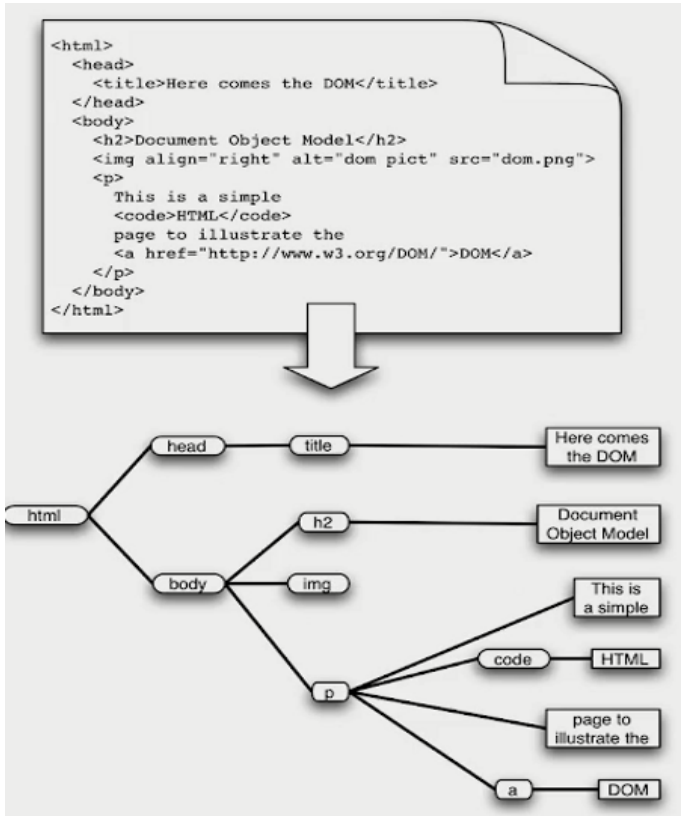
```
import magic
mime = magic.Magic(mime=True)
mime.from_file("test") # 'image/png'
```

Implementation Issues (cont'd)

- **Fetching**
  - Get only the first 10-100 KB per page.
  - Take care to detect and break redirection loops.
  - Soft fail for timeout, server not responding, file not found, and other errors.

More Implementation Issues: Parsing

- HTML has the structure of a DOM (Document Object Model) tree.
- Unfortunately, actual HTML is often incorrect in a strict syntactic sense.
- Crawlers, like browsers, must be robust/forgiving.
- Fortunately, there are tools that can help.
  - E.g. HTML Tidy (`www.html-tidy.org`)



More Implementation Issues: Parsing

- Must pay attention to HTML entities and text encoding (UTF-8, ISO-8859-1, Windows-1252, etc.).
  - **How to detect a file encoding?**
    - HTTP response `charset` in `Content-Type`.
    - Defined in HTML: `<meta http-equiv="Content-Type" content="text/html; charset=utf-8">`
    - Use `file -i` command and convert the file encoding using the `iconv` command or use Python's `chardet` library.
- What to do with a growing number of other formats?
  - Flash, SVG, RSS, AJAX...

More Implementation Issues

- **Static vs. dynamic pages**
  - Is it worth trying to eliminate dynamic pages and only index static pages?
  - Examples:
    - `http://www.census.gov/cgi-bin/gazetteer`
    - `http://informatics.indiana.edu/research/colloquia.asp`
    - `http://www.amazon.com/exec/obidos/subst/home/home.html/002-8332429-6490452`
    - `http://www.imdb.com/Name?Menczer,+Erico`
    - `http://www.imdb.com/name/nm0578801/`
  - Why or why not? How can we tell if a page is dynamic? What about 'spider traps'?
  - What do Google and other search engines do?

More Implementation Issues

- **Relative vs. Absolute URLs**
  - Crawler must translate relative URLs into absolute URLs.
  - Need to obtain Base URL from HTTP header, HTML Meta tag, or else current page path by default.
  - Examples:
    - **Base:** `http://www.cnn.com/linkto`
    - **Relative URL:** `intl.html`
    - **Absolute URL:** `http://www.cnn.com/linkto/intl.html`
    - **Relative URL:** `/US/`
    - **Absolute URL:** `http://www.cnn.com/US/`

More Implementation Issues

- **URL Canonicalization**
  - All of these:
    - `http://www.cnn.com/TECH/`
    - `http://WWW.CNN.COM/TECH/`
    - `http://www.cnn.com:80/TECH/`
    - `http://www.cnn.com/bogus/../TECH/`
  - Are really equivalent to this canonical form:
    - `http://www.cnn.com/TECH/`
  - In order to avoid duplication, the crawler must transform all URLs into canonical form.
  - Definition of "canonical" is arbitrary, e.g.:
    - Could always include port.
    - Or only include port when not default :80.

More on Canonical URLs

- Some transformations are trivial, for example:
  - ✗ `http://informatics.indiana.edu`
  - ✓ `http://informatics.indiana.edu/`
  - ✗ `http://informatics.indiana.edu/index.html#fragment`
  - ✓ `http://informatics.indiana.edu/index.html`
  - ✗ `http://informatics.indiana.edu/dir1/../dir2/`
  - ✓ `http://informatics.indiana.edu/dir2/`
  - ✗ `http://informatics.indiana.edu/%7Efil/`
  - ✓ `http://informatics.indiana.edu/ fil/`
  - ✗ `http://INFORMATICS.INDIANA.EDU/fil/`
  - ✓ `http://informatics.indiana.edu/fil/`

More on Canonical URLs

- Other transformations require heuristic assumptions about the intentions of the author or configuration of the web server:
  - Removing default file name
    - ✓ `http://informatics.indiana.edu/fil/index.html`
    - ✗ `http://informatics.indiana.edu/fil/`
    - This is reasonable in general but would be wrong in this case because the default happens to be `default.asp` instead of `index.html`.
  - Trailing directory
    - ✗ `http://informatics.indiana.edu/fil`
    - ✓ `http://informatics.indiana.edu/fil/`
    - This is correct in this case but how can we be sure in general that there isn't a file named `fil` in the root directory?

More Implementation Issues

- **Spider traps**
  - Misleading sites: indefinite number of pages dynamically generated by server-side scripts.
  - Paths of arbitrary depth created using soft directory links and path rewriting features in HTTP server.
  - `http://example.com/bar/foo/bar/foo/bar/foo/bar/...`
- **Online calendars**

More Implementation Issues: heuristic defensives

- Only heuristic defensive measures:
  - Check URL length: assume spider trap above some threshold, for example 128 characters.
  - Watch for sites with very large number of URLs.
  - Eliminate URLs with non-textual data types.
  - May disable crawling of dynamic pages, if they can be detected.

Bad Actor

BBC Home | Search | Explore the BBC

Low graphics | Accessibility help

BBC NEWS | LIVE BBC NEWS CHANNEL | News services Your news when you want it

News Front Page
World
UK
England
Northern Ireland
Scotland
Wales
Business
Politics
Health
Education
Science & Environment
Technology
Entertainment
Also in the news

Video and Audio

Have Your Say
Magazine
In Pictures
Country Profiles
Special Reports

RELATED BBC SITES

Last Updated: Monday, 6 February 2006, 15:31 GMT

E-mail this to a friend | Printable version

BMW given Google 'death penalty'

Search giant Google has "blacklisted" German car manufacturer BMW for breaching its guidelines.

Investigations by Google found that BMW's German website influenced search results to ensure top ranking when users searched for "used car."

Google has now reduced BMW's page rank to zero, ensuring the company no longer appears at the top.

BMW admitted using so-called "doorway pages" to boost search rankings, but denied any attempt to mislead users.

Some of the suspect pages already appear to have been removed

BMW's activities were revealed in a blog by Google software engineer Matt Cutts.

SEE ALSO:
- Internet firms 'bowed to Beijing' 02 Feb 06 | Americas
- Google shares fall on Wall Street 01 Feb 06 | Business
- Google's communications breakdown 01 Feb 06 | Business
- Why Google in China makes sense 27 Jan 06 | Technology
- The world according to Google 20 Jan 06 | Business
- Google defies US over search data 20 Jan 06 | Technology
- Google taps into search patterns 22 Dec 05 | Technology

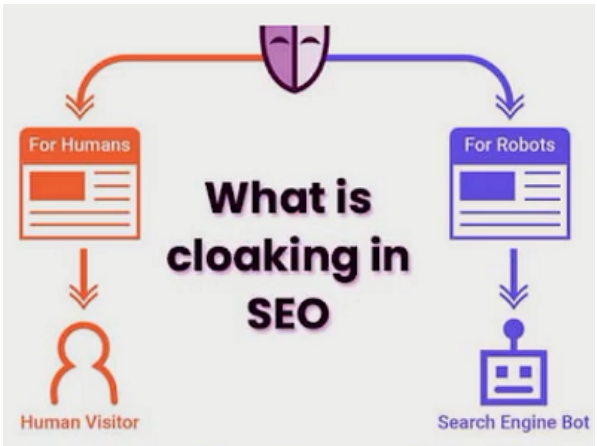
RELATED INTERNET LINKS:
- BMW Germany site
- Google

The BBC is not responsible for the content of external internet sites

Source: http://news.bbc.co.uk/1/hi/technology/4685750.stm

Bad Actors (cont'd)

- Doorway page: Web pages that are created for the deliberate manipulation of search engine indexes (spamdexing).
  - A doorway page will affect the index of a search engine by inserting results for particular phrases while sending visitors to a different page.
  - Doorway pages that redirect visitors without their knowledge use some form of cloaking.
- Cloaking: A *search engine optimization* (SEO) technique in which the content presented to the search engine spider is different from that presented to the user's browser.

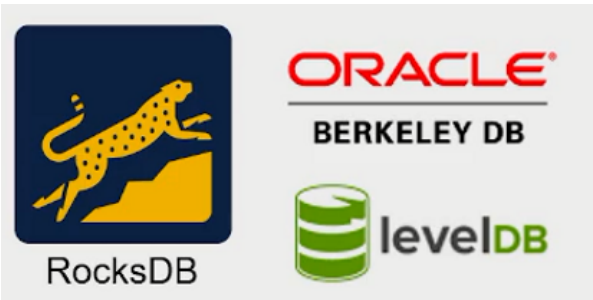


More Implementation Issues

- **Page repository**
  - Naïve: Store each page as a separate file
    - Can map URL to unique filename using a hashing function, e.g., MD5.
    - This generates a huge number of files, which is inefficient from the storage perspective.
  - Better: Combine many pages into a single large file, using some XML markup to separate and identify them
    - Must map URL to {`filename, page_id`}.

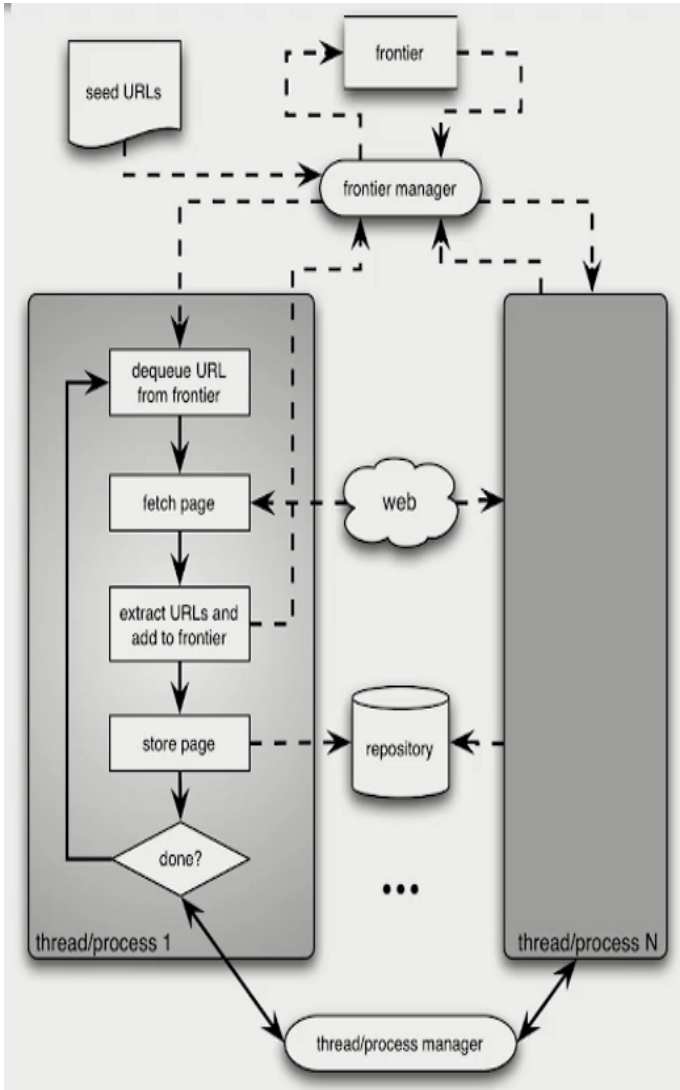
More Implementation Issues

- Database options
  - Any RDBMS – large overhead
  - Light-weight, embedded databases such as Berkeley DB, Google LevelDB, or Facebook RocksDB.



Concurrency

- A crawler incurs several delays:
  - Resolving the host name in the URL to an IP address using DNS.
  - Connecting a socket to the server and sending the request.
  - Receiving the requested page in response.
- Solution: Overlap the above delays by fetching many pages concurrently.



Concurrent Crawlers

- Can use multi-processing or multi-threading.
- Each process or thread works like a sequential crawler, except they share data structures: frontier and repository.
- Shared data structures must be synchronized (locked for concurrent writes).
- Speedup of factor of 5-10 are easy this way.

How would you crawl Wikipedia?

- Wikipedia is a great resource for building a knowledge base (KB) or knowledge graph (KG).
- Google, Facebook, and many other companies use Wikipedia content to enrich their search results.
- How would you crawl Wikipedia?
  - You should NOT.
  - Use Wikipedia dumps: `https://dumps.wikimedia.org/`
  - Farsi Wikipedia dumps: `https://dumps.wikimedia.org/fawiki/`

- Motivation
- Basic crawlers and implementation issues
- **Universal crawlers**
- Preferential (focused and topical) crawlers
- Evaluation
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers
- Crawling in action (wget, curl, httrack, Nutch, Heritrix, and Stormcrawler)

Universal Crawlers

- Support universal search engines, such as Google and Bing.
- Large-scale
- Huge cost (network bandwidth) of crawl is amortized over many queries from users.
- Incremental updates to existing index and other data repositories.

Large-Scale Universal Crawlers

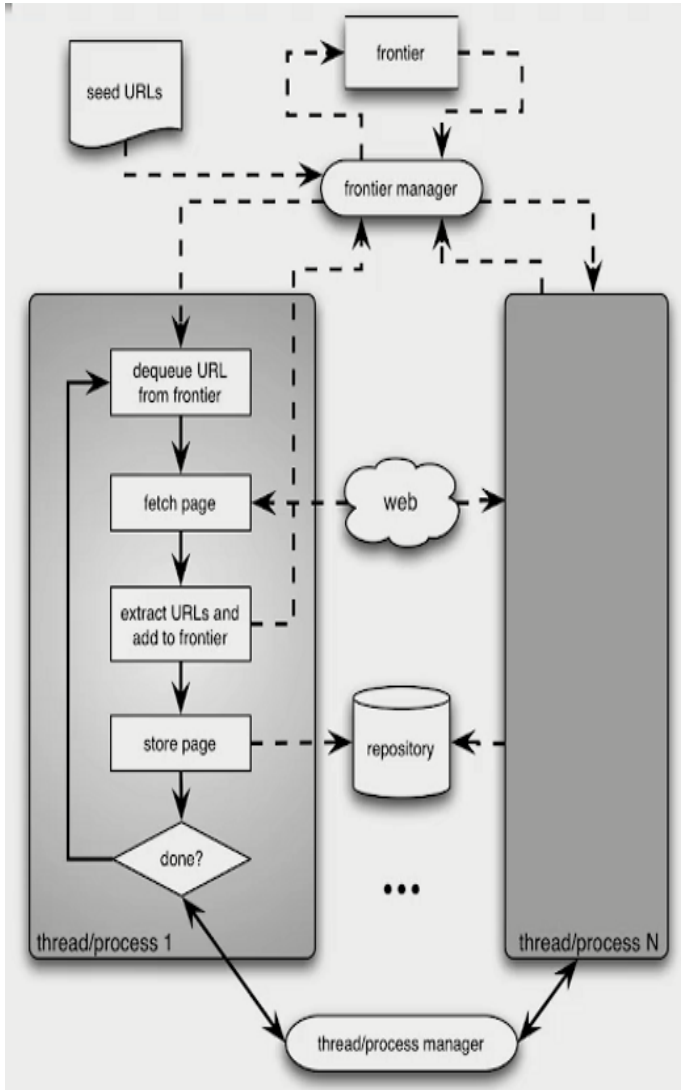
- Two major issues:
  - Performance
    - Need to scale up to billions of pages
  - Policy
    - Need to trade-off coverage, freshness, and bias (e.g., toward `"important"` pages)

Large-Scale Crawlers: Scalability

- Need to minimize overhead of DNS lookups.
- Need to optimize utilization of network bandwidth and disk throughput (I/O is bottleneck).
- Use asynchronous sockets.
  - Multi-processing or multi-threading do not scale up to billions of pages.
  - Non-blocking: Hundreds of network connections open simultaneously.
  - Polling socket to monitor completion of network transfers.

High-Level Architecture of a Scalable Universal Crawler

- Several parallel queues to spread load across servers (keep connections alive).
- DNS server using UDP (less overhead than TCP), large persistent in-memory cache, and prefetching.
- Optimize use of network bandwidth.
- Huge farm of crawl machines.
- Optimize disk I/O throughput.



Universal Crawlers: Policy

- Coverage
  - New pages get added all the time.
  - Can the crawler find every page? **No**.
- Freshness
  - Pages change over time, get removed, etc.
  - How frequently can a crawler revisit?
- Trade-off!
  - Focus on most `"important"` pages (crawler bias)?
  - `"Importance"` is subjective.

Maintaining a "Fresh" Collection

- Universal crawlers are never "done."
- High variance in rate and amount of page changes.
- HTTP headers are notoriously unreliable.
  - `Last-modified`
  - `Expires`
- Solution:
  - Estimate the probability that a previously visited page has changed in the meanwhile.
  - Prioritize by this probability estimate.

HTTP Response Headers (Practical Example)

Curl Command Output

Terminal Output

```
$ curl -s -I www.google.com
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
P3P: "CP=\" This is not a P3P policy! See
g.co/p3phelp for more info.\""
Date: Sun, 02 Mar 2025 10:55:10 GMT
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
Expires: Sun, 02 Mar 2025 10:55:10 GMT
Cache-Control: private
Set-Cookie: (Its a HAsh); expires=Fri,
29-Aug-2025 10:55:10 GMT; path=/;
```

Estimating Page Change Rates

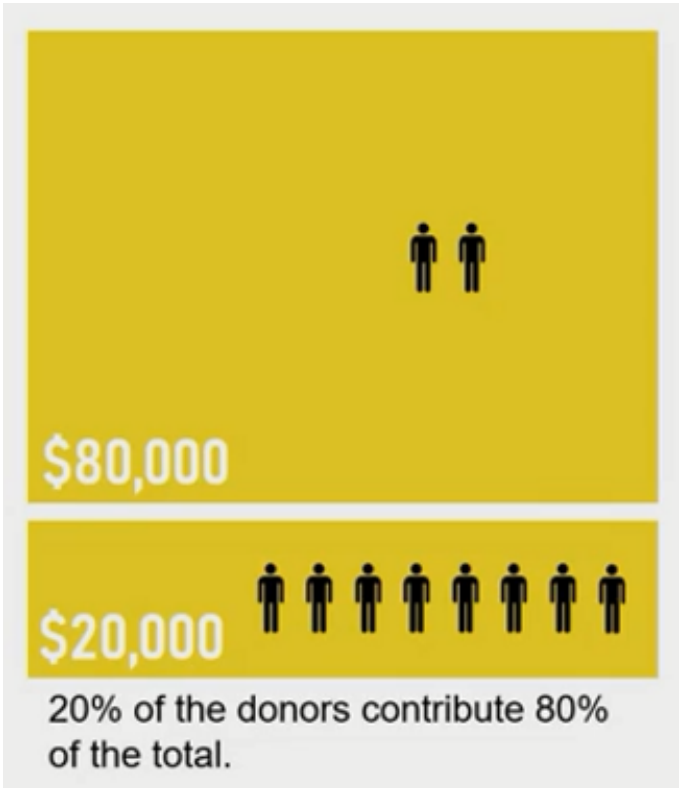
- Algorithms for maintaining a crawl in which most pages are fresher than a specified epoch.
  - **Reading:** J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering", *Computer networks and ISDN systems*, 1998, 161-172.
- **Assumption:** Recent past predicts the future (Ntoulas, Cho & Olston 2004).
  - Frequency of change is not a good predictor.
  - Degree of change is a better predictor.
- **Reading:** A. Ntoulas, J. Cho, and C. Olston, "What's new on the Web? The evolution of the Web from a search engine perspective", *Proceedings of the 13th international conference on World Wide Web*, 2004.

Do we need to crawl the entire Web?

- If we cover too much, it will get stale.
- There is an abundance of pages in the Web.
- For *PageRank*, pages with very low prestige are largely useless.
- What is the goal?
  - General search engines: Pages with high prestige.
  - News portals: Pages that change often.
  - Vertical portals: Pages on some topic.
- What are appropriate priority measures in these cases? Approximations?

Pareto Principle or 80/20 Rule

- The Pareto principle states that for many outcomes, roughly 80% of consequences come from 20% of causes (the "vital few").
- Example: Microsoft noted that by fixing the top 20% of the most-reported bugs, 80% of the related errors and crashes in a given system would be eliminated.
- How would you apply this principle to crawling the web?



Breadth-first Crawlers

- BF crawler tends to crawl high-*PageRank* pages very early.
- Therefore, BF crawler is a good baseline to gauge other crawlers.

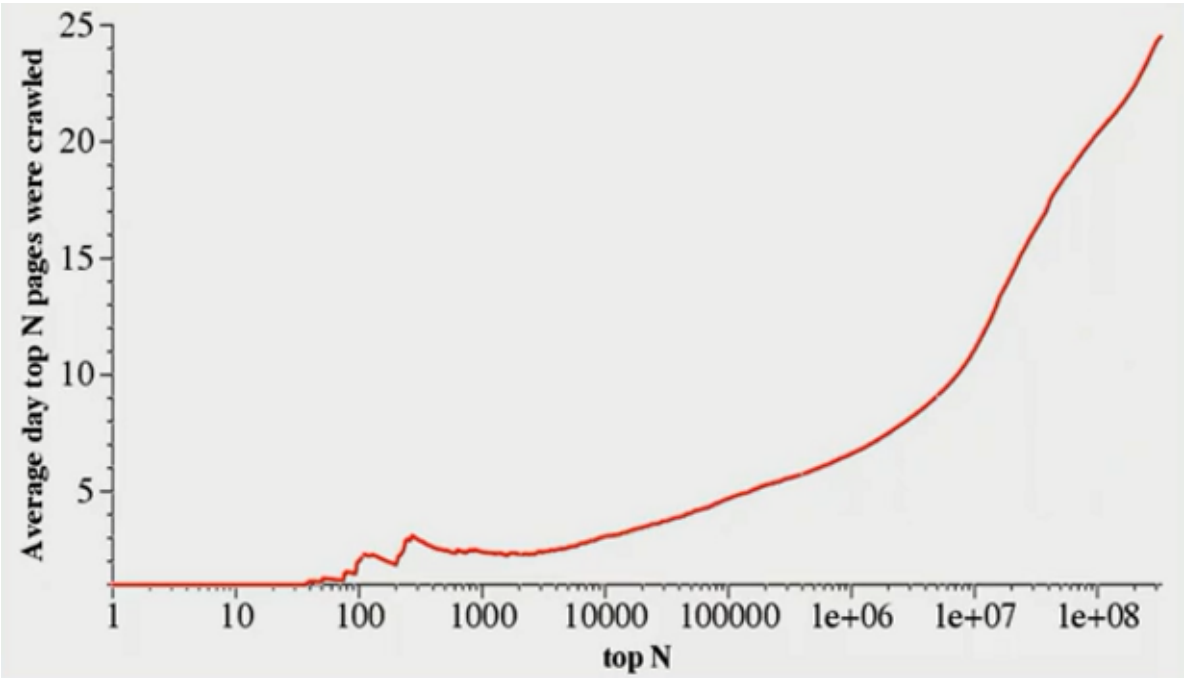


Figure: Najork and Weiner 2001

Bias of Breadth-First Crawlers

- The structure of the Web graph is very different from a random network.
- Power-law distribution of in-degree.
- Therefore, there are hub pages with very high PR and many incoming links.
- These are attractors: you cannot avoid them!

Crawler Etiquette

Crawler Etiquette

- Avoid crawling online ads to generate more profit.
- Do not crawl and mirror competitors' websites.
  - This is a violation of copyright law.
- Identify the crawler properly through `User-Agent`.
  - Example: `Mozilla/5.0 (compatible; Googlebot/2.1; http://www.google.com/bot.html)`
- Crawler Politeness: Respect implicit and explicit politeness considerations.
- Web server etiquette:
  - Use of AI to detect impolite crawling.
    - Use captcha.
  - Avoid using cloaking (remember the BMW story).

Explicit and Implicit Politeness

- Crawler Politeness: Respect implicit and explicit politeness considerations.
  - Explicit politeness: Specifications from webmasters on what portions of a site can be crawled.
    - E.g., robots.txt
  - Implicit politeness: Even with no specification, avoid hitting any site too often.

Robots.txt

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

Robots.txt Example

- No robot should visit any URL starting with `/yoursite/temp/`, except the robot called **"searchengine"**:

Robots.txt Rules

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

User-agent:  searchengine
Disallow:
```

Filters and robots.txt

- Filters – regular expressions for URLs to be crawled/not.
- Once a robots.txt file is fetched from a site, need not fetch it repeatedly.
  - Doing so burns bandwidth, hits web server.
- Cache robots.txt files.

Sitemap

- A protocol to inform search engines about URLs on a website that are available for crawling.
  - It is supposed to help and guide the crawler.
- An XML file that lists the URLs for a site.
- Allows webmasters to include additional information about each URL:
  - Last updated
  - How often it changes
  - How important it is in relation to other URLs of the site
- This allows search engines to crawl the site more efficiently and to find URLs that may be isolated from the rest of the site's content.
- A URL inclusion protocol and complements robots.txt, a URL exclusion protocol.
  - Usually, the sitemap is linked within robots.txt.

Sitemap – Example 1

XML Sitemap Example

```
<?xml version="1.0" encoding="utf-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/
sitemap/0.9"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.sitemaps.org/
schemas/sitemap/0.9
http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"

<url>
<loc>http://example.com/</loc>
<lastmod>2006-11-18</lastmod>
<changefreq>daily</changefreq>
<priority>0.8</priority>
</url>
</urlset>
```

Sitemap – Example 2

XML Sitemap Index Example

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org
/schemas/sitemap/0.9">

<sitemap>
<loc>http://www.example.com/sitemap1.xml.gz</loc>
<lastmod>2014-10-01T18:23:17+00:00</lastmod>
</sitemap>

</sitemapindex>
```

Politeness – Challenges

Even if we restrict only one thread to fetch from a host, can hit it repeatedly.

Politeness – Challenges

Even if we restrict only one thread to fetch from a host, can hit it repeatedly.

- Common heuristic: insert time gap between successive requests to a host that is $>>$ time for most recent fetch from that host.

Crawler in Action

GNU Wget

- A free software package for retrieving files using HTTP, HTTPS, FTP, and FTPS, the most widely used Internet protocols.
- It is a non-interactive **command line** tool.
  - It may easily be called from scripts, cron jobs, terminals without X-Windows support, etc.
  - It's available for Linux, Mac, and Windows.
- Latest version:
  - 1.x: **Very stable**.
  - 2.x: Still **buggy** and unstable.
- This is my favorite tool

Some Useful Command Line Options

- **Logging and Input File Options**
  - `-i file` / `--input-file=file`
- **Download Options**
  - `-t number` / `--tries=number`
  - `-O file` / `--output-document=file`
  - `-c` / `--continue`
  - `-T seconds` / `--timeout=seconds`
  - `--limit-rate=amount`
  - `-w seconds` / `--wait=seconds`
  - `--random-wait`
  - `-Q quota` / `--quota=quota`
- **Directory Options**
  - `-x` / `--force-directories`

Some Useful Command Line Options

- **HTTP Options**
  - `-E` / `--adjust-extension`
  - `--save-cookies file`
  - `--load-cookies file`
  - `--header=header-line`
  - `--compression=type`
  - `--max-redirect=number`
  - `-U agent-string` / `--user-agent=agent-string`
  - `--content-disposition`
  - `--trust-server-names`
- **Recursive Retrieval Options**
  - `-r` / `--recursive`
  - `-l depth` / `--level=depth`
  - `-k` / `--convert-links`
- **Recursive Accept/Reject Options**
  - `-A acclist` / `--accept acclist`
  - `-R rejlist` / `--reject rejlist`
  - `-np` / `--no-parent`

Recursive Crawling using wget

Command

```
$ wget --recursive \
--no-parent \
--no-host-directories \
--reject "index.html" \
--user-agent="BigDataBot" \
https://ilam.ac.ir/
```