Web Crawling

Based on the slides by Filippo Menczer @Indiana University School of Informatics in *Web Data Mining* by Bing Liu

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Crawler ethics and conflicts

Google Search: spears

Web Images Groups News Froogle more »

Google

spears Search Advanced Search Preferences

Web Results **1** - **10** of about **9,440,000** for **spears** [definition]. (**0.14** seconds)

News results for **spears** - View today's top stories
Knee Injury Closes **Spears**' Onyx Hotel - Billboard - 1 hour ago
Britney **Spears**' tour is canceled - San Diego Union Tribune - 7 hours ago
As fall approaches, **Spears** may start to smell Curious - Houston Chronicle - Jun 14, 2004

Britney **Spears** :: The Official Web Site
The Official Web Site of Britney **Spears**. Your official source for all things Britney. ...
Remember, proceeds benefit the Britney **Spears** Foundation. ...
www.britney**spears**.com/ - 41k - Jun 14, 2004 - Cached - Similar pages

Britney **Spears** - britney.com - Jive Records
iTunes. Real/Rhapsody. Napster. Under 11.
www.britney.com/ - 10k - Cached - Similar pages

Britney **Spears** Portal - pics, lyrics, MP3s and more!
Britney **Spears** pics, lyrics, MP3s, news, gossip, fan sites, forums, and much more!
Britney **Spears** Portal, ...); ');. Britney **Spears** Portal. ...
www.britney-**spears**-portal.com/ - 25k - Cached - Similar pages

Britney **Spears** guide to Semiconductor Physics: semiconductor ...
Britney **Spears** lectures on semiconductor physics, radiative and non-radiative transitions, edge emitting lasers and VCSELs. ...
britney**spears**.ac/lasers.htm - 13k - Cached - Similar pages

BritneySpears.org: Your online guide to Britney!
A comprehensive Britney **Spears** fansite which pays tribute to Britney with the most active message board, daily news, many pictures, desktop media and more. ...
www.britney**spears**.org/ - 78k - Jun 14, 2004 - Cached - Similar pages

Britney-**Spears**.To You! - The Britney **Spears** Community
Britney **Spears** : biography, discography, musics, real, mp3, videos, pictures, clips, guestbook, www board, free page, search engine, links and more. ...
www.britney-**spears**.to/ - 9k - Cached - Similar pages

The Mystery of Britney's Breasts
www.liquidgeneration.com/poptoons/britneys_breasts.asp - 2k - Cached - Similar pages

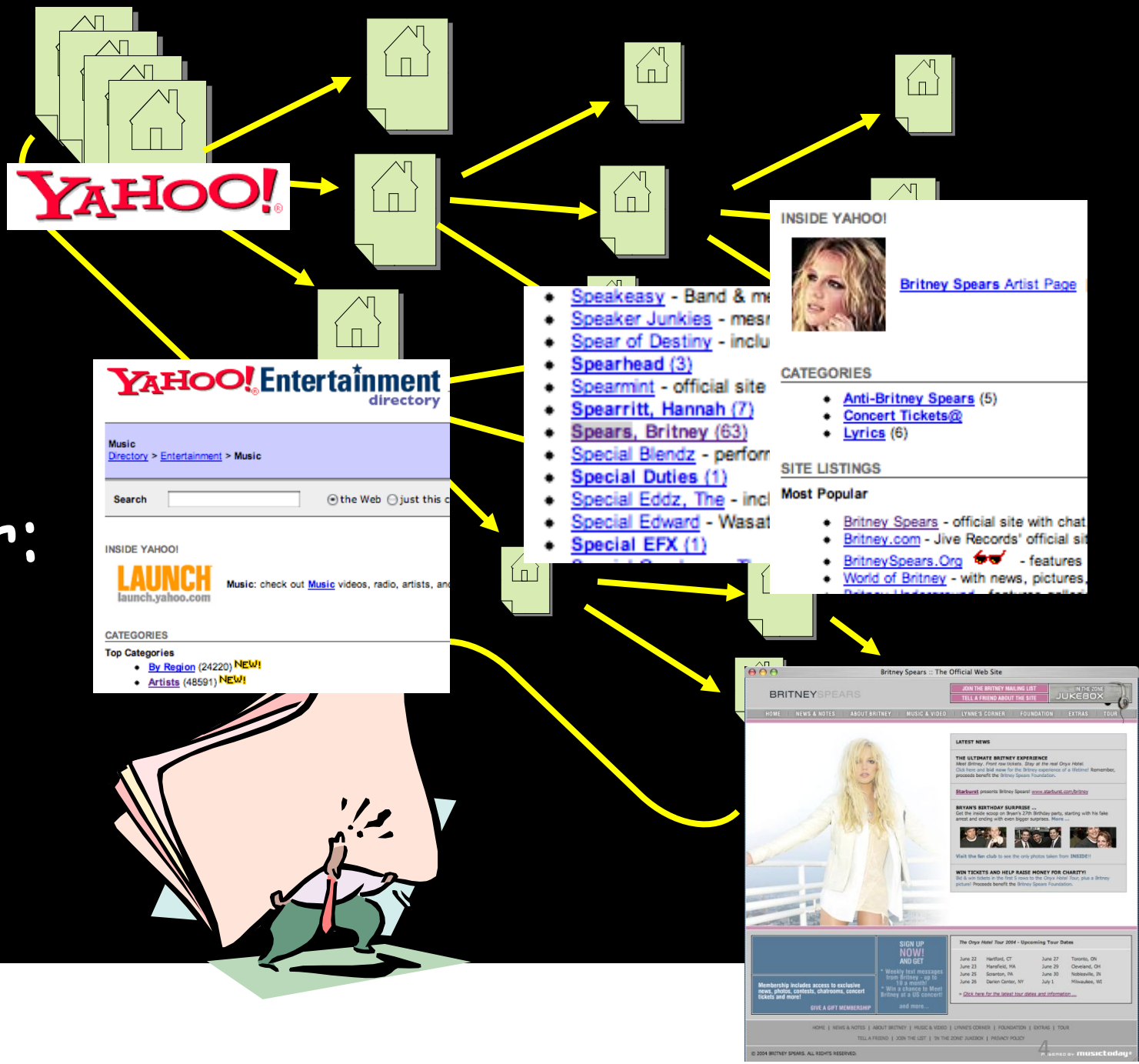
Britney **Spears** spelling correction
The data below shows some of the misspellings detected by our spelling correction system for the query [britney **spears**], and the count of how many different ...
www.google.com/jobs/britney.html - 40k - Cached - Similar pages

Britney **Spears** pictures news music Britney **Spears** lyrics
Britney **Spears** pictures mp3 sites gallery photos images music fun games chat lyrics. ...
Britney **Spears** Forum Come see what is inside the Britney **Spears** forum! ...
www.britney-**spears**.com/ - 42k - Jun 14, 2004 - Cached - Similar pages

Britney **Spears** Zone - Your Guide to Britney Pictures and News
Britney **Spears**, Britney **Spears**, Britney **Spears**, ... Britney **Spears**, ...
www.britneyzone.com/ - 101k - Jun 14, 2004 - Cached - Similar pages

Q: How does a search engine know that all these pages contain the query terms?
A: Because all of those pages have been crawled

Crawler: basic idea



Many names

- Crawler
- Spider
- Robot (or bot)
- Web agent
- Wanderer, worm, …
- And famous instances: googlebot, scooter, slurp, msnbot, …

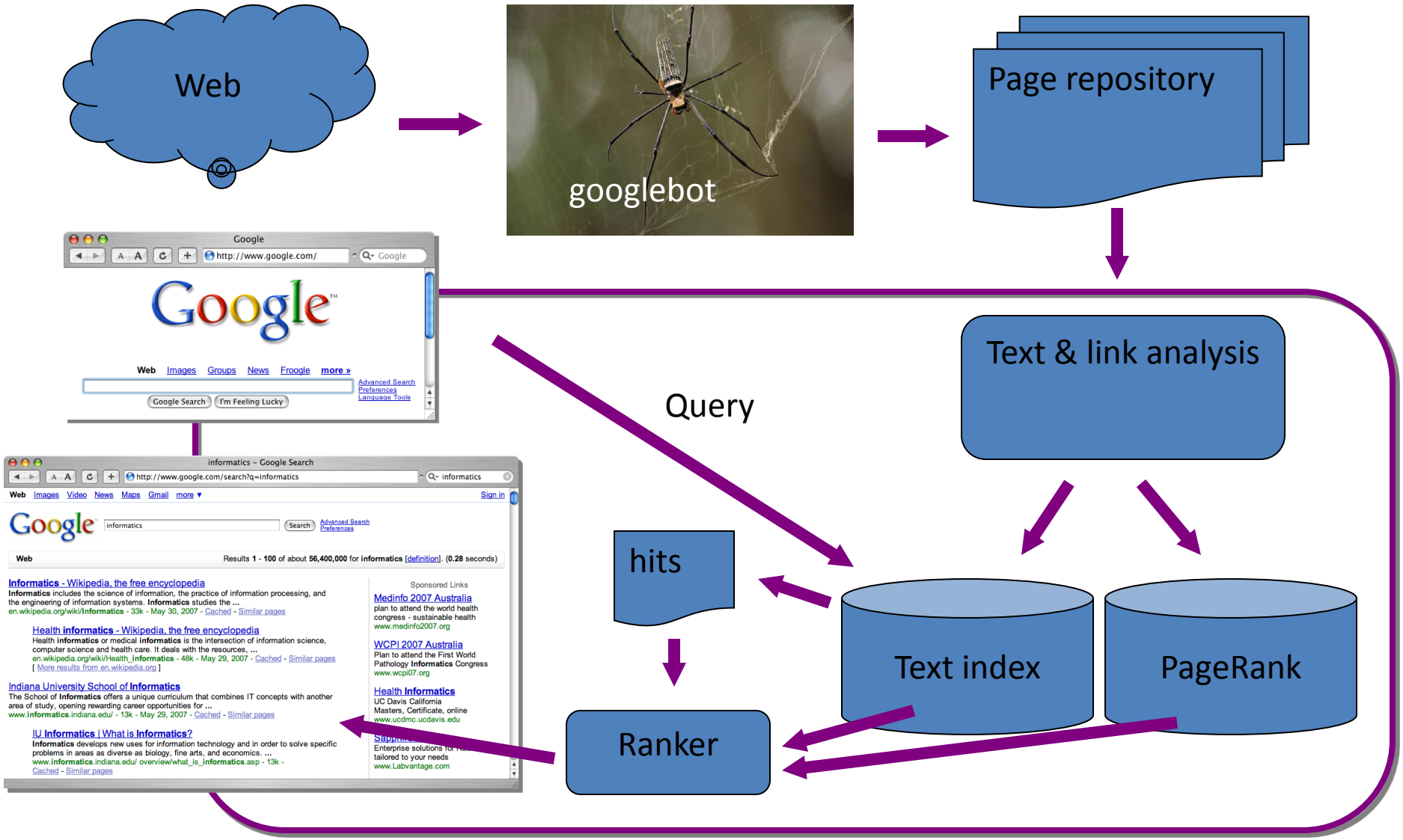
Googlebot & you

```
tcsh — ⌘1
homer:~% more /var/log/httpd/access_log
129.217.55.111 - - [11/Sep/2004:04:36:24 -0500] "GET /~fil/Thanksgiving/1999/Pages/Image10.html HTTP/1.0" 200 302
84.135.208.173 - - [11/Sep/2004:04:40:57 -0500] "GET /~fil/Max/2000/Fall/November/ HTTP/1.1" 404 320
80.100.20.198 - - [11/Sep/2004:04:41:40 -0500] "GET /~fil/Max/2000/Fall/November/ HTTP/1.0" 404 308
64.68.82.182 - - [11/Sep/2004:04:41:51 -0500] "GET /robots.txt HTTP/1.0" 404 290
62.39.213.35 - - [11/Sep/2004:04:41:52 -0500] "GET /~fil/Max/2000/Fall/November/ HTTP/1.0" 404 308
64.68.82.182 - - [11/Sep/2004:04:41:52 -0500] "GET /network/network.map HTTP/1.0" 200 3544
129.217.55.111 - - [11/Sep/2004:04:41:58 -0500] "GET /~fil/Max/2003/Fall/Fall-Pages/Image3.html HTTP/1.0" 200 491
129.217.55.111 - - [11/Sep/2004:04:42:01 -0500] "GET /~fil/Max/2002/Spring/Spring-Pages/Image6.html HTTP/1.0" 200 495
129.217.55.111 - - [11/Sep/2004:04:42:03 -0500] "GET /~fil/Max/2002/Europe02/Crans-Montana/ HTTP/1.0" 200 6361
129.217.55.111 - - [11/Sep/2004:04:42:36 -0500] "GET /~fil/Vacation/Europe02/Venezia/Pages/Image12.html HTTP/1.0" 200 352
129.217.55.111 - - [11/Sep/2004:04:43:01 -0500] "GET /~fil/Thanksgiving/1999/Pages/Image9.html HTTP/1.0" 200 301
129.217.55.111 - - [11/Sep/2004:04:43:43 -0500] "GET /~fil/Max/2003/Fall/Fall-Pages/Image2.html HTTP/1.0" 200 485
129.217.55.111 - - [11/Sep/2004:04:43:45 -0500] "GET /~fil/Max/2002/Spring/Spring-Pages/Image5.html HTTP/1.0" 200 498
129.217.55.111 - - [11/Sep/2004:04:43:48 -0500] "GET /~fil/Max/2002/Europe02/Bologna/ HTTP/1.0" 200 2469
129.217.55.111 - - [11/Sep/2004:04:44:14 -0500] "GET /~fil/Vacation/Europe02/Venezia/Pages/Image11.html HTTP/1.0" 200 352
129.217.55.111 - - [11/Sep/2004:04:44:49 -0500] "GET /~fil/Thanksgiving/1999/Pages/Image8.html HTTP/1.0" 200 301
129.217.55.111 - - [11/Sep/2004:04:45:30 -0500] "GET /~fil/Max/2003/Fall/Fall-Pages/Image1.html HTTP/1.0" 200 485
129.217.55.111 - - [11/Sep/2004:04:45:31 -0500] "GET /~fil/Max/2002/Spring/Spring-Pages/Image4.html HTTP/1.0" 200 501
129.217.55.111 - - [11/Sep/2004:04:45:57 -0500] "GET /~fil/Vacation/Europe02/Venezia/Pages/Image10.html HTTP/1.0" 200 352
129.217.55.111 - - [11/Sep/2004:04:46:25 -0500] "GET /~fil/Thanksgiving/1999/Pages/Image7.html HTTP/1.0" 200 301
129.217.55.111 - - [11/Sep/2004:04:50:27 -0500] "GET /~fil/Max/2003/Fall/Fall-Pages/Image0.html HTTP/1.0" 200 495
129.217.55.111 - - [11/Sep/2004:04:50:30 -0500] "GET /~fil/Max/2002/Spring/Spring-Pages/Image3.html HTTP/1.0" 200 501
129.217.55.111 - - [11/Sep/2004:04:50:59 -0500] "GET /~fil/Vacation/Europe02/Venezia/Pages/Image9.html HTTP/1.0" 200 318
129.217.55.111 - - [11/Sep/2004:04:51:32 -0500] "GET /~fil/Thanksgiving/1999/Pages/Image6.html HTTP/1.0" 200 301
129.217.55.111 - - [11/Sep/2004:04:52:40 -0500] "GET /~fil/Max/2002/Spring/Spring-Pages/Image2.html HTTP/1.0" 200 522
homer:~% host 64.68.82.182
182.82.68.64.in-addr.arpa domain name pointer crawler14.googlebot.com.
homer:~%
```

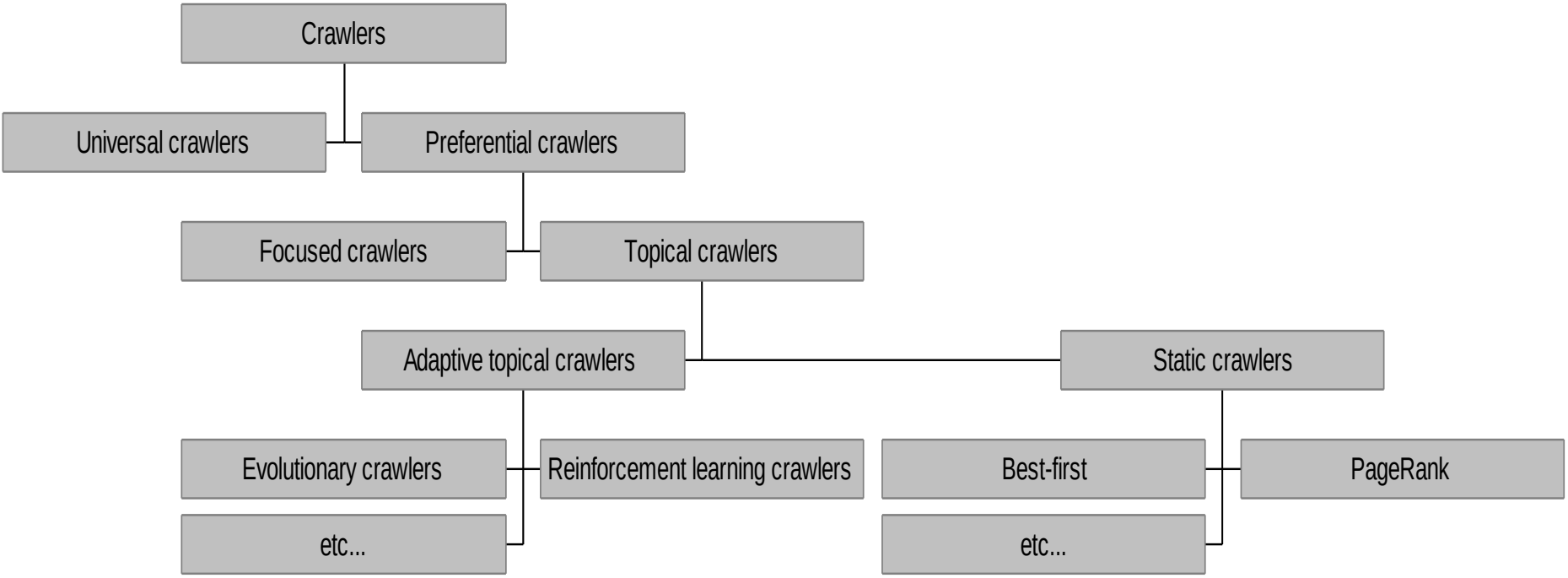
Motivation for crawlers

- Support universal search engines (Google, Yahoo, MSN/Windows Live, Ask, etc.)
- Vertical (specialized) search engines, e.g. news, shopping, papers, recipes, reviews, etc.
- Business intelligence: keep track of potential competitors, partners
- Monitor Web sites of interest
- Evil: harvest emails for spamming, phishing…
- … Can you think of some others?…

A crawler within a search engine



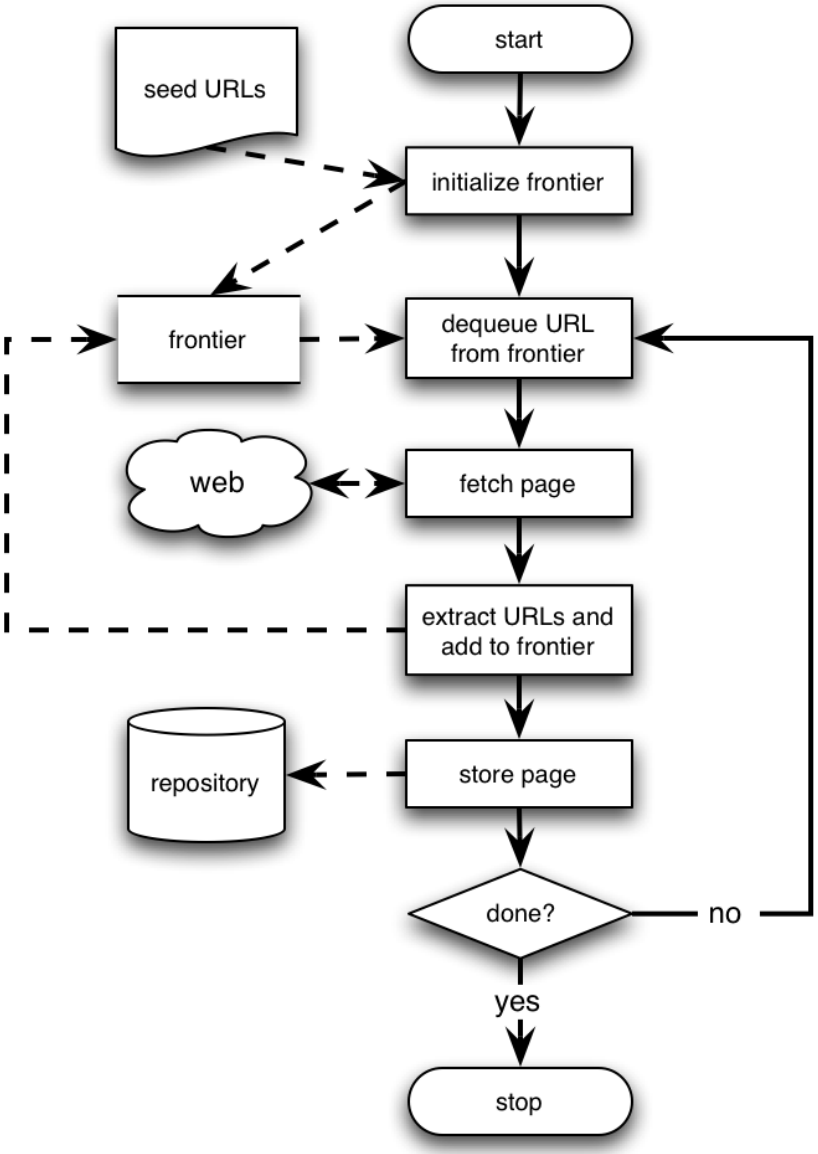
One taxonomy of crawlers



- Many other criteria could be used:
  - Incremental, Interactive, Concurrent, Etc.

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Crawler ethics and conflicts

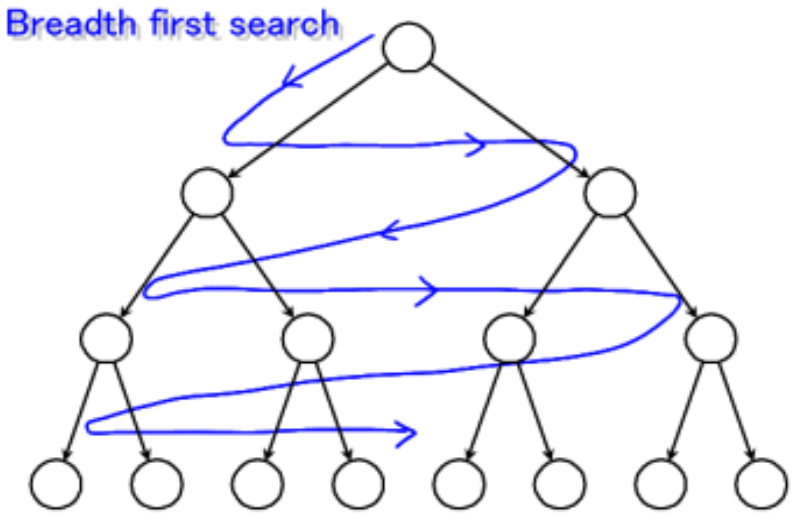


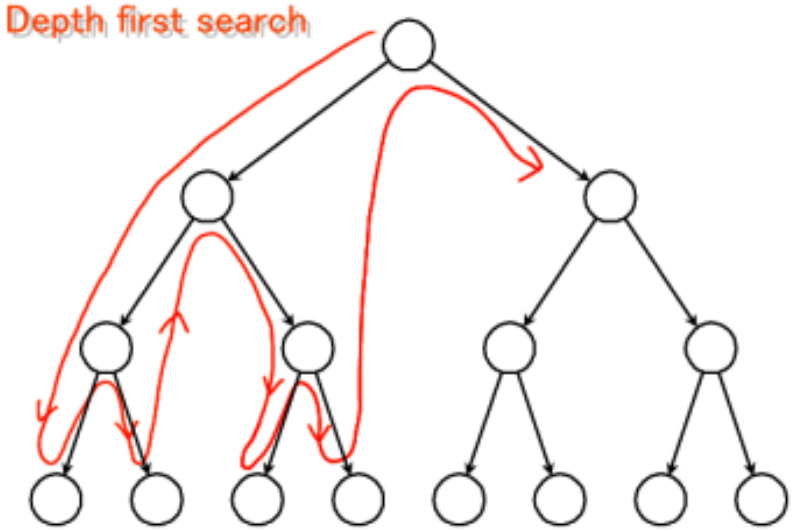
Basic crawlers

- 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

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”)





A basic crawler in Perl

- Queue: a FIFO list (shift and push)

```
my @frontier = read_seeds($file);
while (@frontier && $tot < $max) {
    my $next_link = shift @frontier;
    my $page = fetch($next_link);
    add_to_index($page);
    my @links = extract_links($page, $next_link);
    push @frontier, process(@links);
}
```

- A workable example

Implementation issues

- Don’t want to fetch same page twice!
  - Keep lookup table (hash) of visited pages
  - What if not visited but in 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 (MIME) headers, but overhead of extra Internet requests

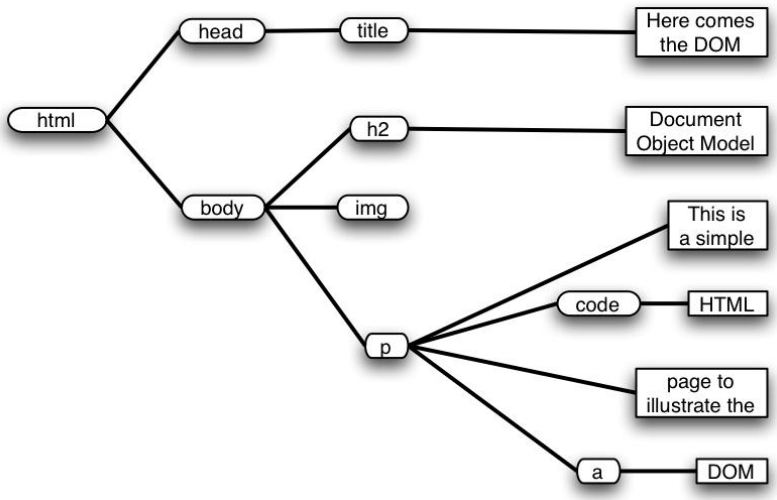
More implementation issues

- 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. tidy.sourceforge.net
- Must pay attention to HTML entities and unicode in text
- What to do with a growing number of other formats?
  - Flash, SVG, RSS, AJAX…

```
<html>
  <head>
    <title>Here comes the DOM</title>
  </head>
  <body>
    <h2>Document Object Model</h2>
    <img align="right" alt="dom pict" src="dom.png">
    <p>
      This is a simple
      <code>HTML</code>
      page to illustrate the
      <a href="http://www.w3.org/DOM/">DOM</a>
    </p>
  </body>
</html>
```



More implementation issues

- Stop words
  - Noise words that do not carry meaning should be eliminated ("stopped") before they are indexed
  - E.g. in English: AND, THE, A, AT, OR, ON, FOR, etc…
  - Typically syntactic markers
  - Typically the most common terms
  - Typically kept in a negative dictionary
    - 10–1,000 elements
    - E.g. http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words
  - Parser can detect these right away and disregard them

More implementation issues

Conflation and thesauri

- Idea: improve recall by merging words with same meaning

1. We want to ignore superficial morphological features, thus merge semantically similar tokens
   - {student, study, studying, studious} => studi
2. We can also conflate synonyms into a single form using a thesaurus
   - 30-50% smaller index
   - Doing this in both pages and queries allows to retrieve pages about ‘automobile’ when user asks for ‘car’
   - Thesaurus can be implemented as a hash table

More implementation issues

- Stemming
  - Morphological conflation based on rewrite rules
  - Language dependent!
  - Porter stemmer very popular for English
    - http://www.tartarus.org/~martin/PorterStemmer/
    - Context-sensitive grammar rules, eg:
      - “IES” except (“EIES” or “AIES”) --> “Y”
    - Versions in Perl, C, Java, Python, C#, Ruby, PHP, etc.
  - Porter has also developed Snowball, a language to create stemming algorithms in any language
    - http://snowball.tartarus.org/
    - Ex. Perl modules: Lingua::Stem and Lingua::Stem::Snowball

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, or 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 transformation 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 assumption about the intentions of the author or configuration of the Web server:

1. 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'
2. 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 dir?

More implementation issues

- Spider traps
  - Misleading sites: indefinite number of pages dynamically generated by CGI scripts
  - Paths of arbitrary depth created using soft directory links and path rewriting features in HTTP server
  - 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 can detect

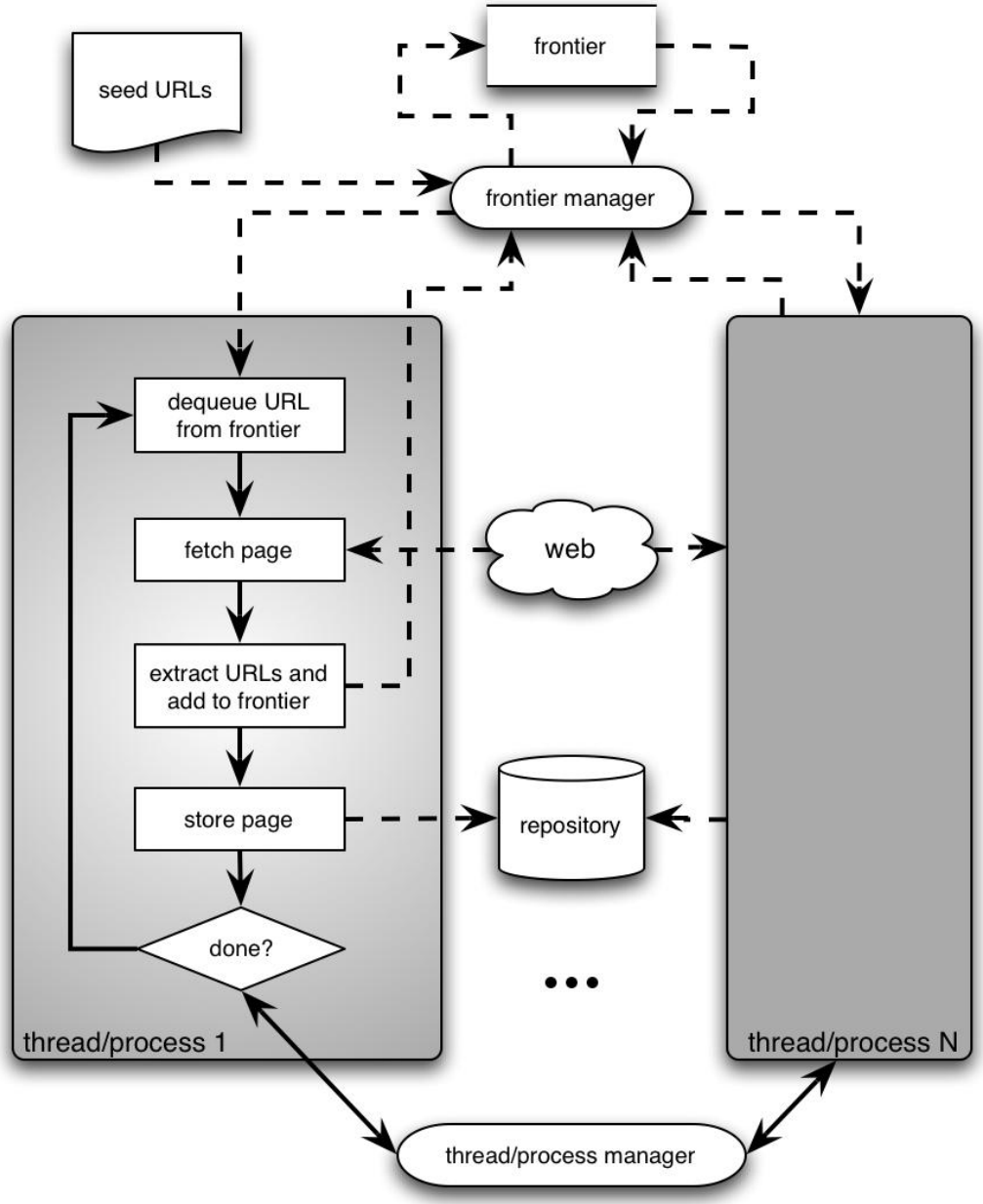
More implementation issues

- 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}
  - Database options
    - Any RDBMS -- large overhead
    - Light-weight, embedded databases such as Berkeley DB

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

Architecture of a concurrent crawler



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

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Crawler ethics and conflicts

Universal crawlers

- Support universal search engines
- 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

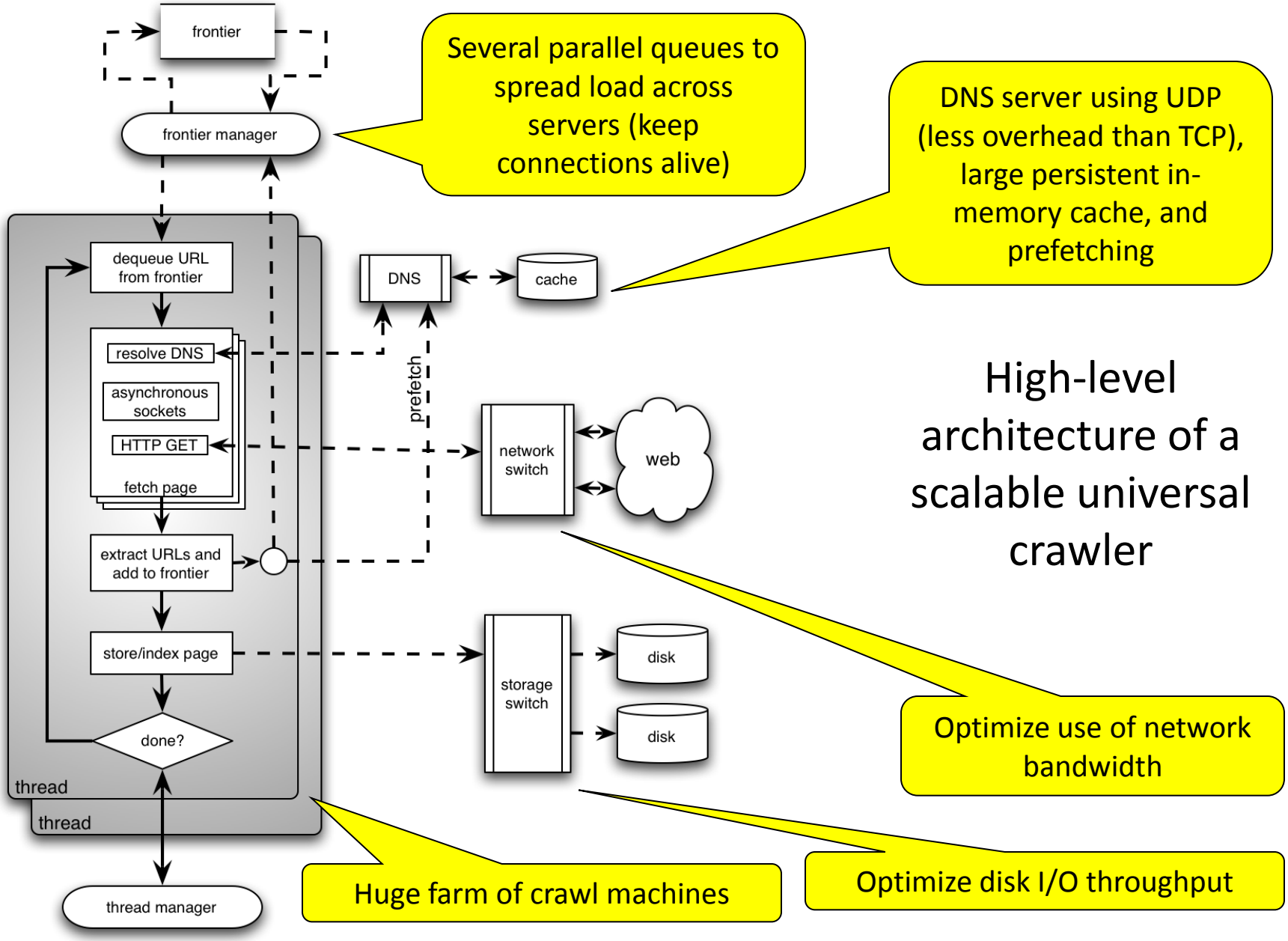
- Two major issues:

1. Performance
   - Need to scale up to billions of pages
2. 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

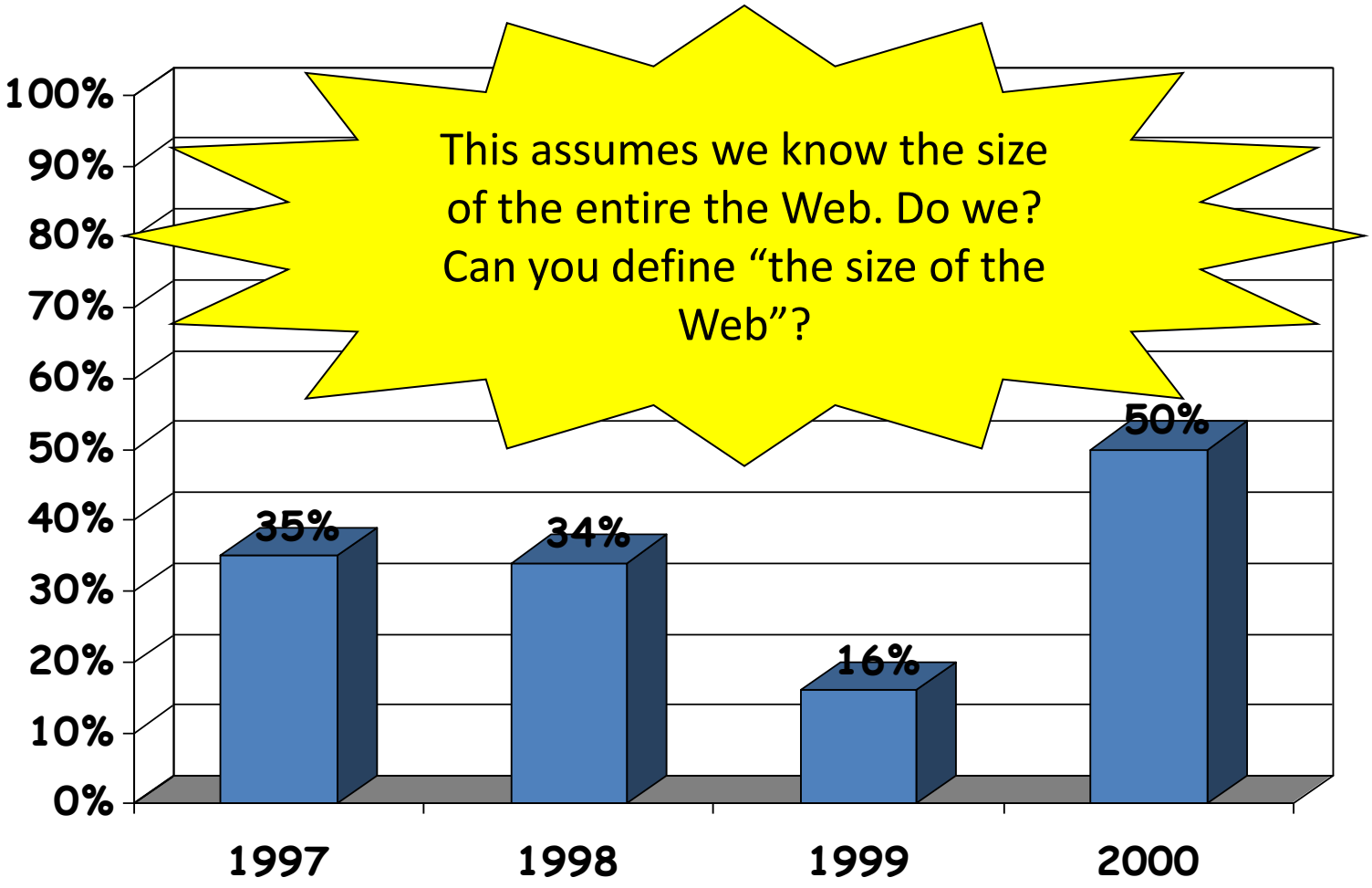
High-level architecture of a scalable universal crawler



Universal crawlers: Policy

- Coverage
  - New pages get added all the time
  - Can the crawler find every page?
- 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

Web coverage by search engine crawlers



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

Estimating page change rates

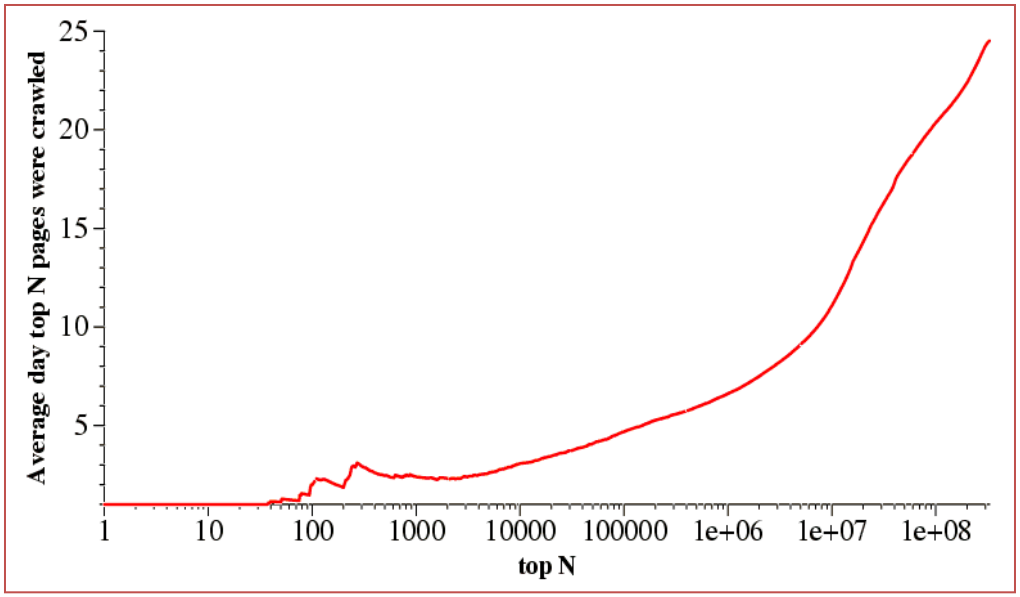
- Algorithms for maintaining a crawl in which most pages are fresher than a specified epoch
  - Brewington & Cybenko; Cho, Garcia-Molina & Page
- Assumption: recent past predicts the future (Ntoulas, Cho & Olston 2004)
  - Frequency of change not a good predictor
  - Degree of change is a better predictor

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?

Breadth-first crawlers

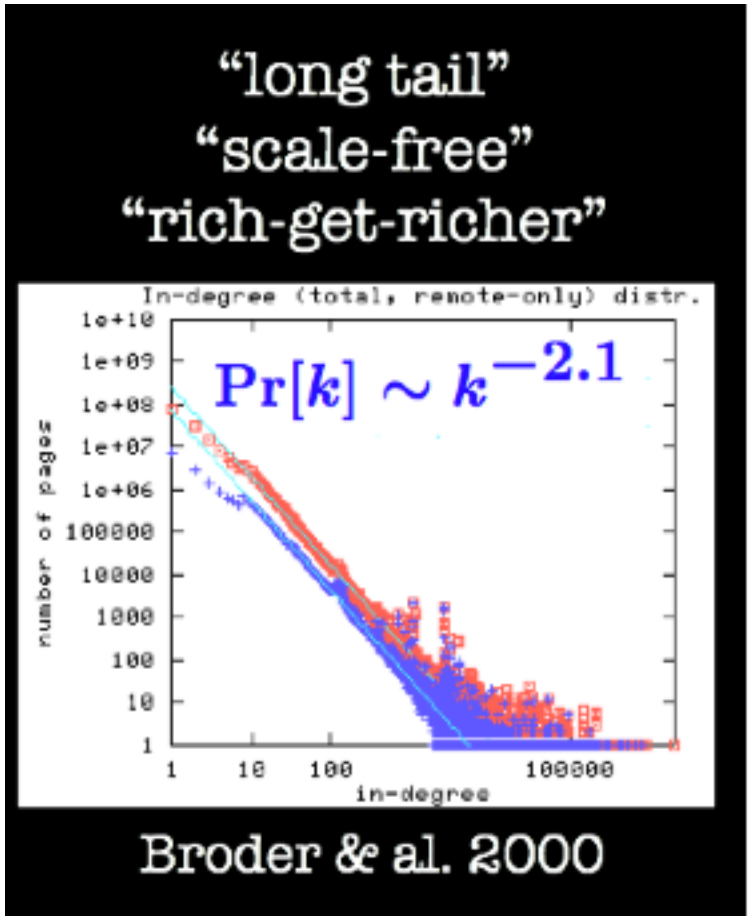
- BF crawler tends to crawl high-PageRank pages very early
- Therefore, BF crawler is a good baseline to gauge other crawlers
- But why is this so?



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!



Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Crawler ethics and conflicts

Crawler ethics and conflicts

- Crawlers can cause trouble, even unwillingly, if not properly designed to be “polite” and “ethical”
- For example, sending too many requests in rapid succession to a single server can amount to a Denial of Service (DoS) attack!
  - Server administrator and users will be upset
  - Crawler developer/admin IP address may be blacklisted

Crawler etiquette (important!)

- Identify yourself
  - Use ‘User-Agent’ HTTP header to identify crawler, website with description of crawler and contact information for crawler developer
  - Use ‘From’ HTTP header to specify crawler developer email
  - Do not disguise crawler as a browser by using their ‘User-Agent’ string
- Always check that HTTP requests are successful, and in case of error, use HTTP error code to determine and immediately address problem
- Pay attention to anything that may lead to too many requests to any one server, even unwillingly, e.g.:
  - redirection loops
  - spider traps

Crawler etiquette (important!)

- Spread the load, do not overwhelm a server
  - Make sure that no more than some max. number of requests to any single server per unit time, say < 1/second
- Honor the Robot Exclusion Protocol
  - A server can specify which parts of its document tree any crawler is or is not allowed to crawl by a file named 'robots.txt' placed in the HTTP root directory, e.g. http://www.indiana.edu/robots.txt
  - Crawler should always check, parse, and obey this file before sending any requests to a server
  - More info at:
    - http://www.google.com/robots.txt
    - http://www.robotstxt.org/wc/exclusion.html

More on robot exclusion

- Make sure URLs are canonical before checking against robots.txt
- Avoid fetching robots.txt for each request to a server by caching its policy as relevant to this crawler
- Let’s look at some examples to understand the protocol…

www.apple.com/robots.txt

```
# robots.txt for http://www.apple.com/

User-agent: *
Disallow:
```

All crawlers…

…can go anywhere!

www.microsoft.com/robots.txt

```
# Robots.txt file for http://www.microsoft.com

User-agent: *
Disallow: /canada/Library/mnp/2/aspx/
Disallow: /communities/bin.aspx
Disallow: /communities/eventdetails.mspx
Disallow: /communities/blogs/PortalResults.mspx
Disallow: /communities/rss.aspx
Disallow: /downloads/Browse.aspx
Disallow: /downloads/info.aspx
Disallow: /france/formation/centres/planning.asp
Disallow: /france/mnp_utility.mspx
Disallow: /germany/library/images/mnp/
Disallow: /germany/mnp_utility.mspx
Disallow: /ie/ie40/
Disallow: /info/customerror.htm
Disallow: /info/smart404.asp
Disallow: /intlkb/
Disallow: /isapi/
#etc…
```

All crawlers…

…are not allowed in these paths…

www.springer.com/robots.txt

```
# Robots.txt for http://www.springer.com (fragment)

User-agent: Googlebot
Disallow: /chl/*
Disallow: /uk/*
Disallow: /italy/*
Disallow: /france/*

User-agent: slurp
Disallow:
Crawl-delay: 2

User-agent: MSNBot
Disallow:
Crawl-delay: 2

User-agent: scooter
Disallow:

# all others
User-agent: *
Disallow: /
```

Google crawler is allowed everywhere except these paths

Yahoo and MSN/Windows Live are allowed everywhere but should slow down

AltaVista has no limits

Everyone else keep off!

More crawler ethics issues

- Is compliance with robot exclusion a matter of law?
  - No! Compliance is voluntary, but if you do not comply, you may be blocked
  - Someone (unsuccessfully) sued Internet Archive over a robots.txt related issue
- Some crawlers disguise themselves
  - Using false User-Agent
  - Randomizing access frequency to look like a human/browser
  - Example: click fraud for ads

More crawler ethics issues

- Servers can disguise themselves, too
  - Cloaking: present different content based on User-Agent
  - E.g. stuff keywords on version of page shown to search engine crawler
  - Search engines do not look kindly on this type of "spamdexing" and remove from their index sites that perform such abuse
    - Case of bmw.de made the news

Gray areas for crawler ethics

- If you write a crawler that unwillingly follows links to ads, are you just being careless, or are you violating terms of service, or are you violating the law by defrauding advertisers?
  - Is non-compliance with Google's robots.txt in this case equivalent to click fraud?
- If you write a browser extension that performs some useful service, should you comply with robot exclusion?