Chapter 8: Web Crawling

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers

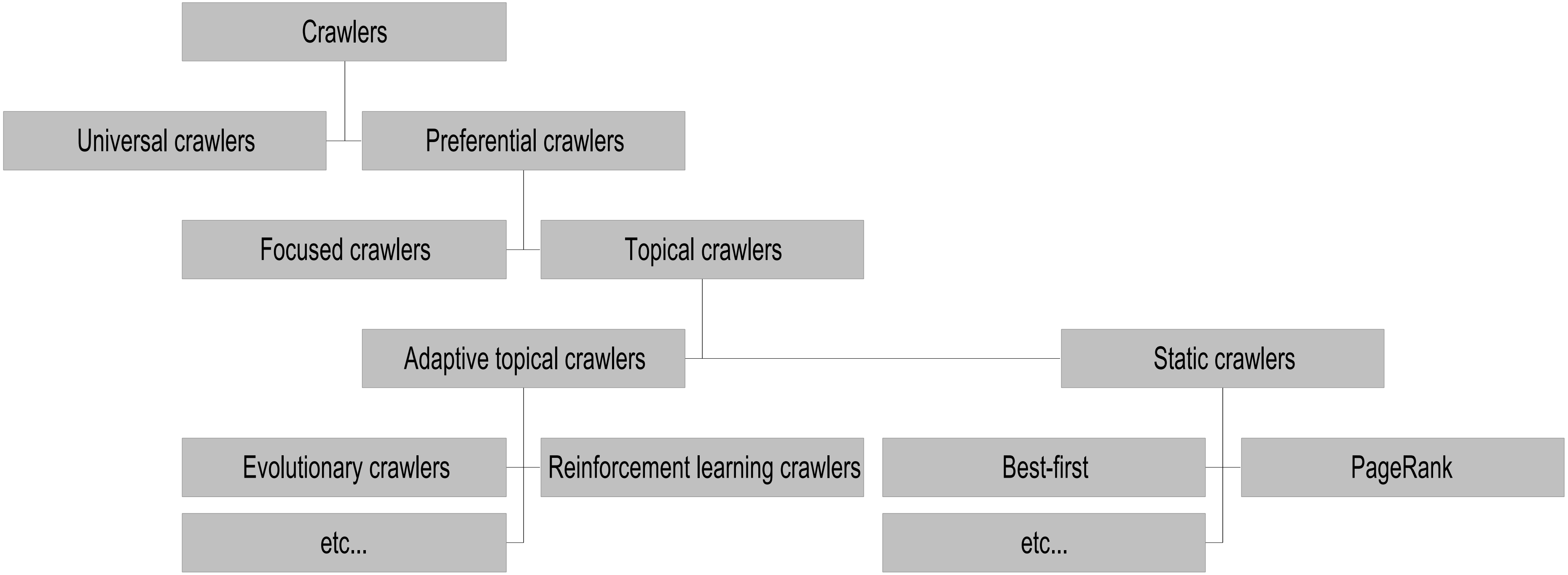
Many names

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

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?…

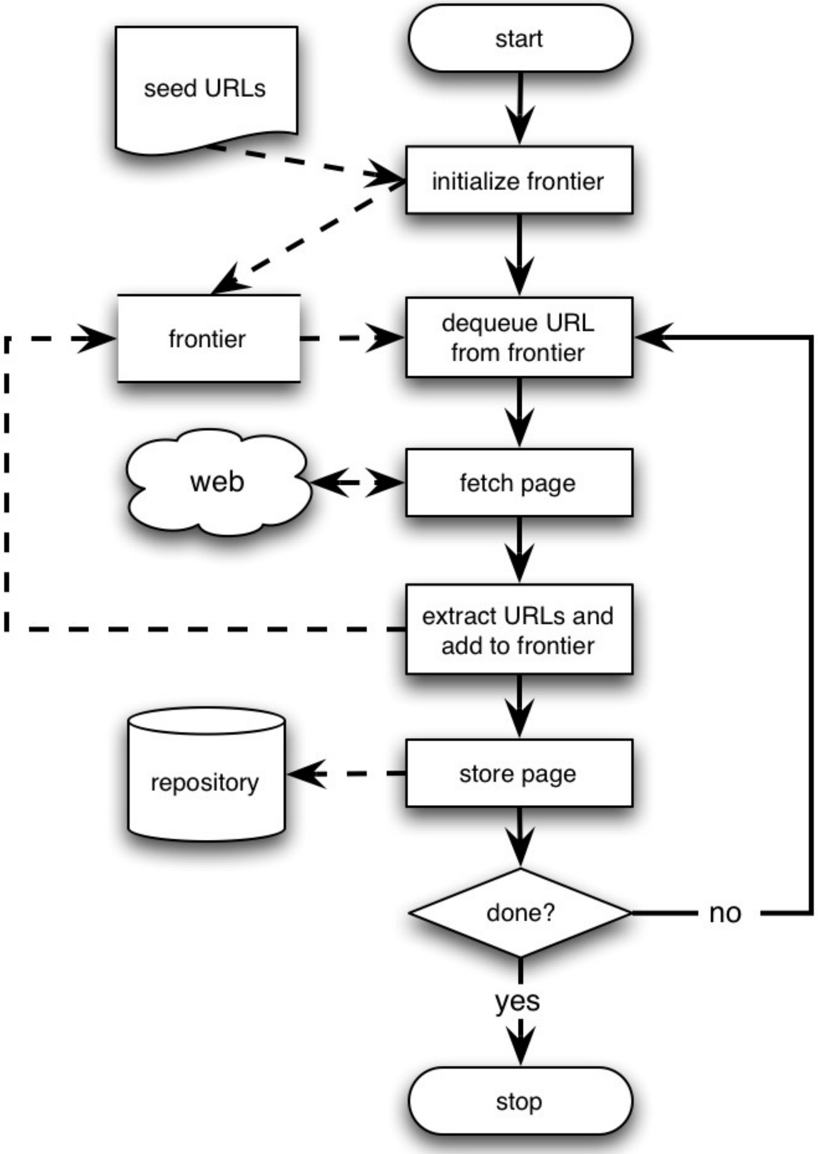
One taxonomy of crawlers



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

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers

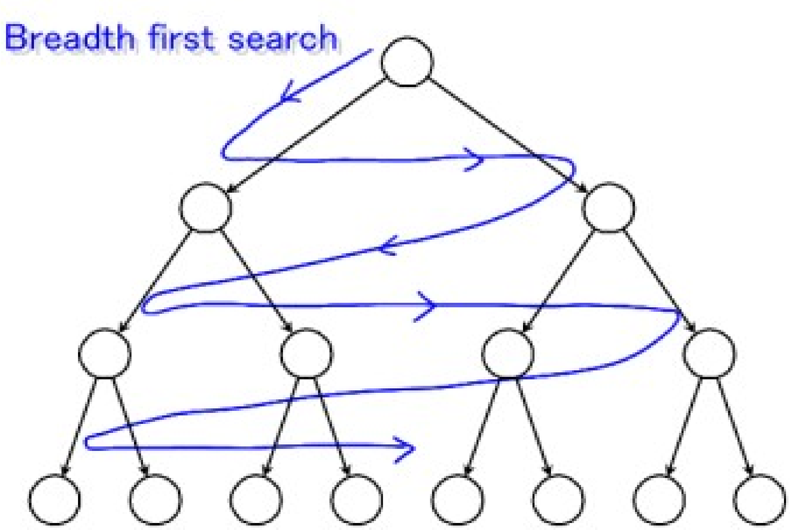


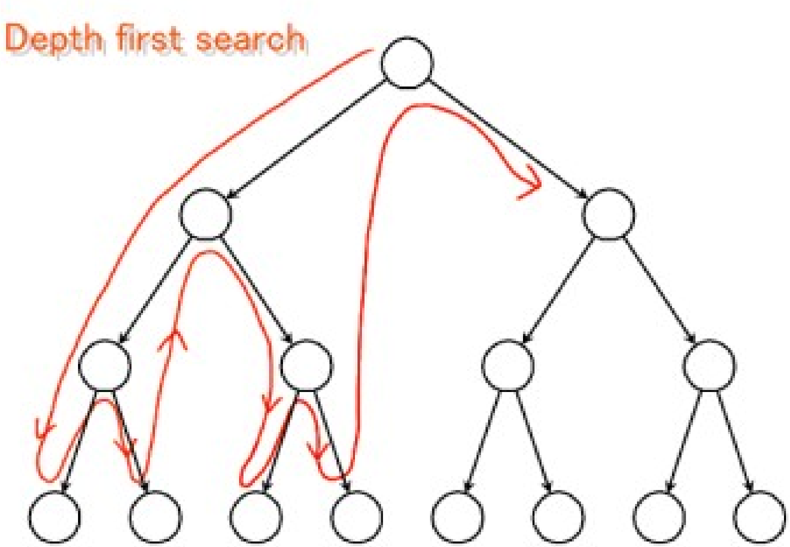
Basic crawlers

- 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

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);
}
```

Implementation issues

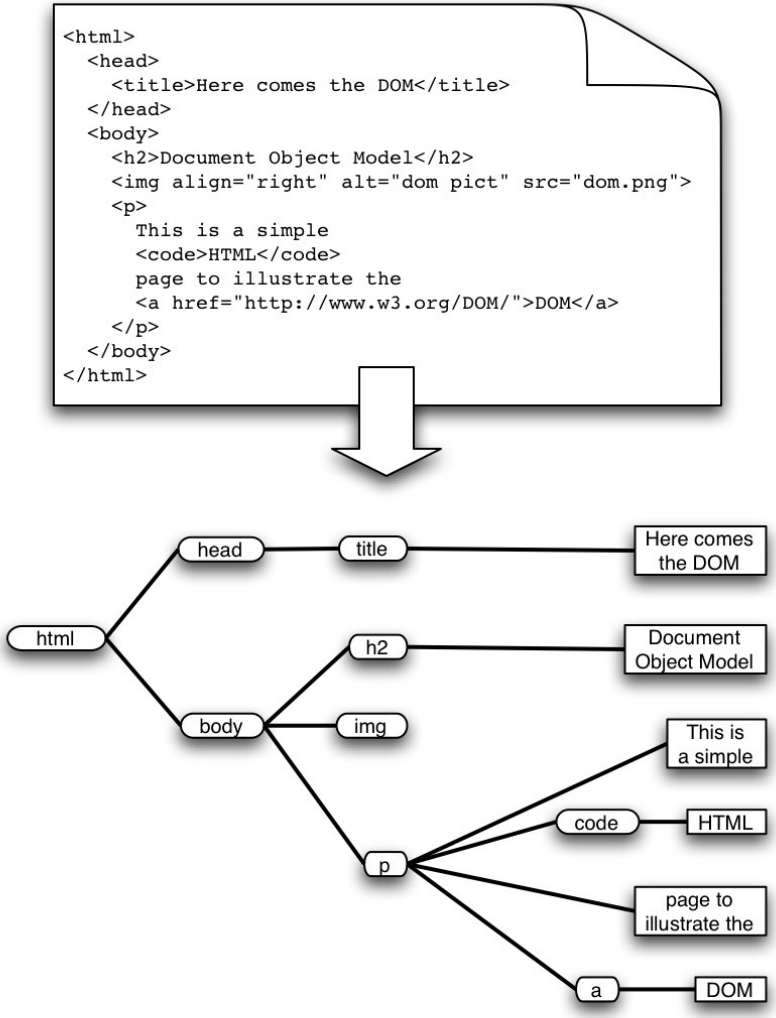
- Don’t want to fetch same page twice!
  - Keep lookup table (hash) of visited pages
  - What if not visited but in frontier already?
- The frontier grows very fast!
  - May need to prioritize for large crawls
- Fetcher must be robust!
  - Don’t crash if download fails
  - Timeout mechanism
- Determine 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

3. We want to ignore superficial morphological features, thus merge semantically similar tokens
   - {student, study, studying, studious} => studi
4. 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:

- 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 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

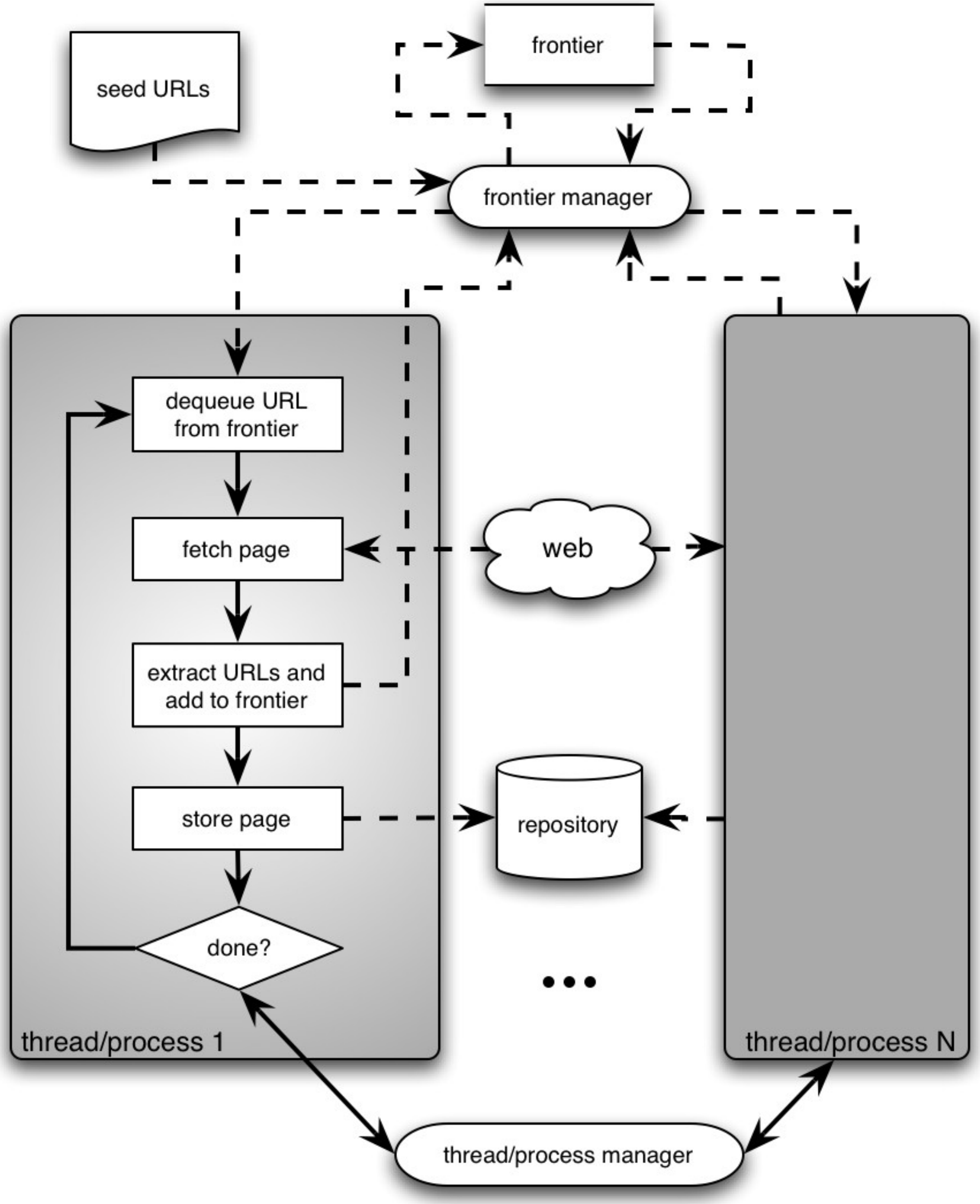
More implementation issues

- 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
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers

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

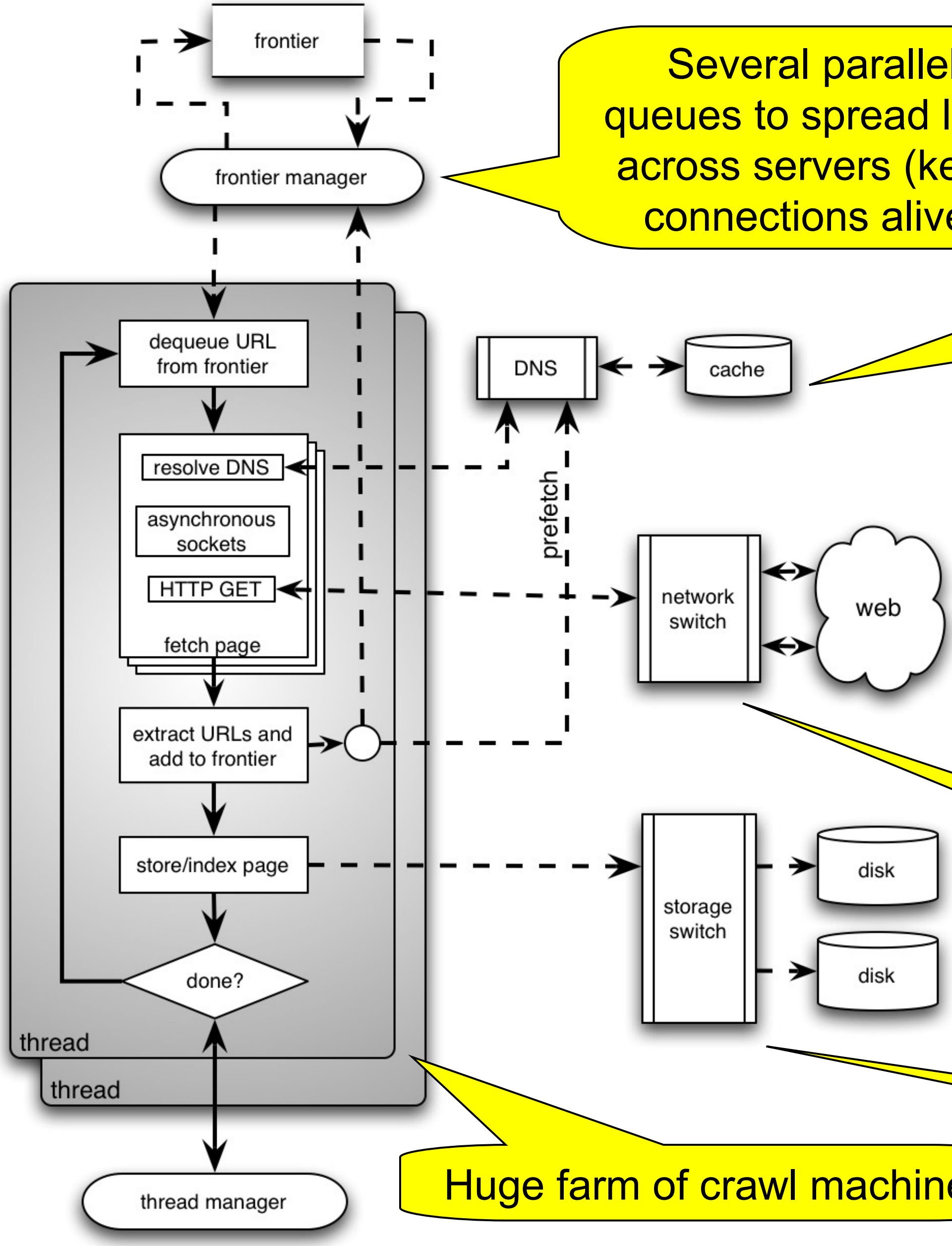
Large-scale universal crawlers

- Two major issues:
- 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

Optimize disk I/O throughput

Huge farm of crawl machines

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

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?

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers

Preferential crawlers

- Assume we can estimate for each page an importance measure, $I(p)$
- Want to visit pages in order of decreasing $I(p)$
- Maintain the frontier as a priority queue sorted by $I(p)$
- Possible figures of merit:
  - Precision ~
    $| p: \text{crawled}(p) \ \& \ I(p) > \text{threshold} \ | \ / \ | \ p: \text{crawled}(p) \ |$
  - Recall ~
    $| p: \text{crawled}(p) \ \& \ I(p) > \text{threshold} \ | \ / \ | \ p: I(p) > \text{threshold} \ |$

Preferential crawlers

- Selective bias toward some pages, eg. most “relevant”/topical, closest to seeds, most popular/largest PageRank, unknown servers, highest rate/amount of change, etc…
- <u>Focused crawlers</u>
  - Supervised learning: classifier based on labeled examples
- <u>Topical crawlers</u>
  - Best-first search based on similarity(topic, parent)
  - Adaptive crawlers
    - Reinforcement learning
    - Evolutionary algorithms/artificial life

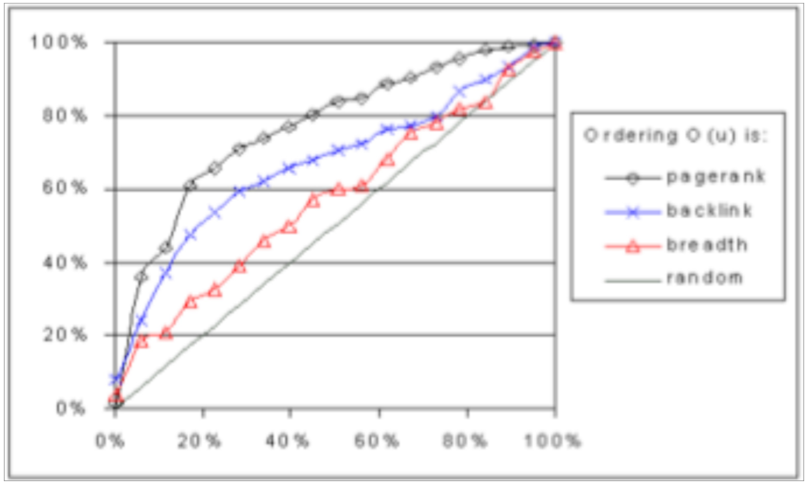
Preferential crawling algorithms: Examples

- Breadth-First
  - Exhaustively visit all links in order encountered
- Best-*N*-First
  - Priority queue sorted by similarity, explore top N at a time
  - Variants: DOM context, hub scores
- PageRank
  - Priority queue sorted by keywords, PageRank
- SharkSearch
  - Priority queue sorted by combination of similarity, anchor text, similarity of parent, etc. (powerful cousin of FishSearch)
- InfoSpiders
  - Adaptive distributed algorithm using an evolving population of learning agents

Preferential crawlers: Examples

- For I(p) = PageRank (estimated based on pages crawled so far), we can find high-PR pages faster than a breadth-first crawler (Cho, Garcia-Molina & Page 1998)

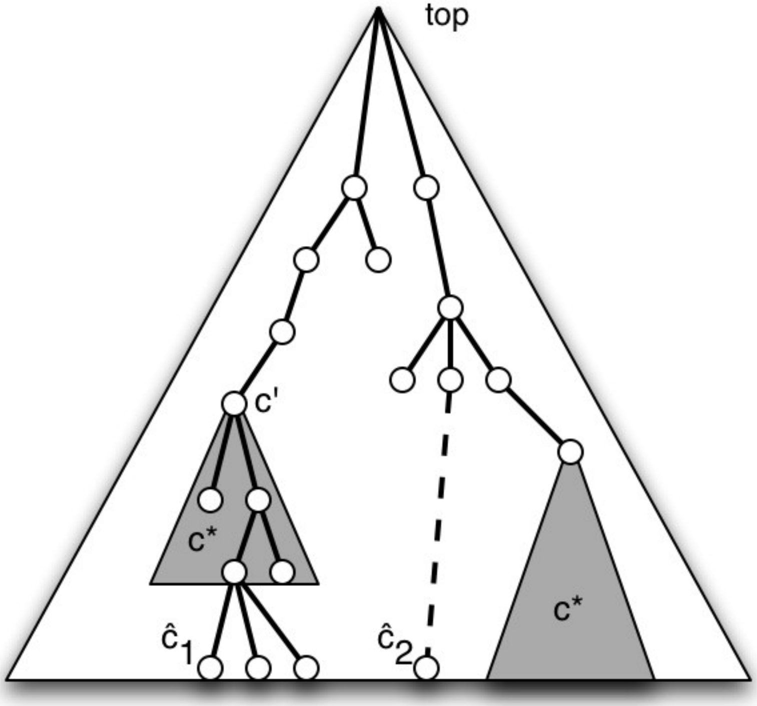
Recall



Crawl size

Focused crawlers: Basic idea

- Naïve-Bayes classifier based on example pages in desired topic, c*
- Score(p) = Pr(c*|p)
  - Soft focus: frontier is priority queue using page score
  - Hard focus:
    - Find best leaf ĉ for p
    - If an ancestor c' of ĉ is in c* then add links from p to frontier, else discard
  - Soft and hard focus work equally well empirically

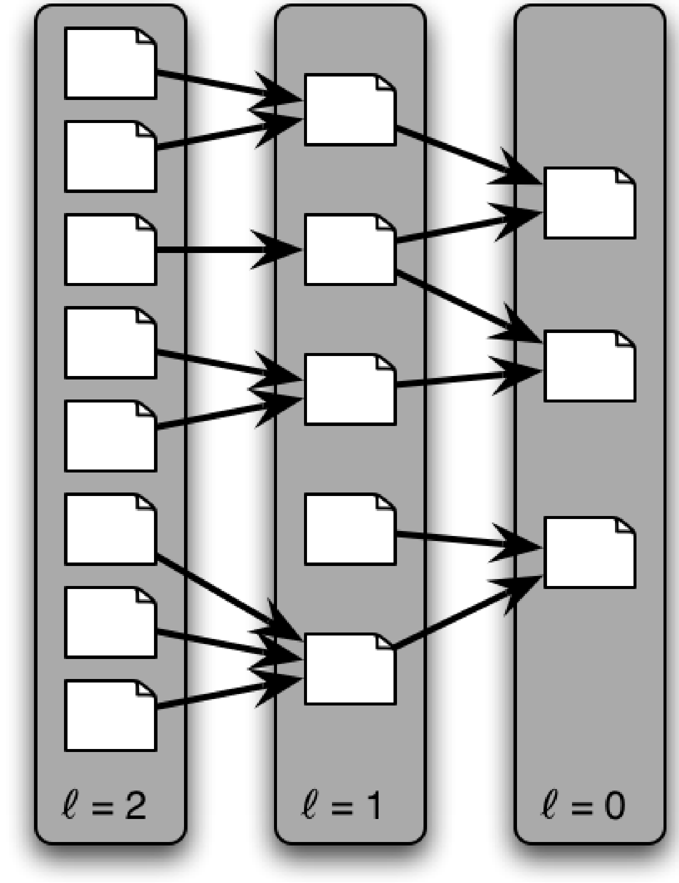


Example: Open Directory

Focused crawlers

- Can have multiple topics with as many classifiers, with scores appropriately combined (Chakrabarti et al. 1999)
- Can use a distiller to find topical hubs periodically, and add these to the frontier
- Can accelerate with the use of a critic (Chakrabarti et al. 2002)
- Can use alternative classifier algorithms to naïve-Bayes, e.g. SVM and neural nets have reportedly performed better (Pant & Srinivasan 2005)

Context-focused crawlers



Context graph

- Same idea, but multiple classes (and classifiers) based on link distance from relevant targets
  - ℓ=0 is topic of interest
  - ℓ=1 link to topic of interest
  - Etc.
- Initially needs a back-crawl from seeds (or known targets) to train classifiers to estimate distance
- Links in frontier prioritized based on estimated distance from targets
- Outperforms standard focused crawler empirically

Topical crawlers

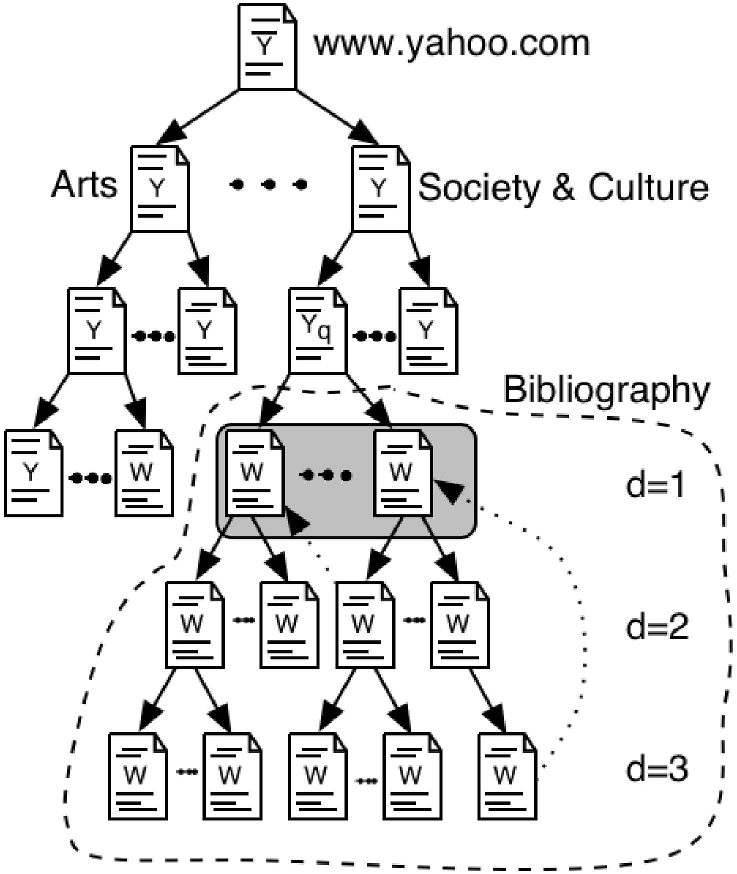
- All we have is a topic (query, description, keywords) and a set of seed pages (not necessarily relevant)
- No labeled examples
- Must predict relevance of unvisited links to prioritize
- Original idea: Menczer 1997, Menczer & Belew 1998

Topical locality

- Topical locality is a necessary condition for a topical crawler to work, and for surfing to be a worthwhile activity for humans
- Links must encode semantic information, i.e. say something about neighbor pages, not be random
- It is also a sufficient condition if we start from “good” seed pages
- Indeed we know that Web topical locality is strong :
  - Indirectly (crawlers work and people surf the Web)
  - From direct measurements (Davison 2000; Menczer 2004, 2005)

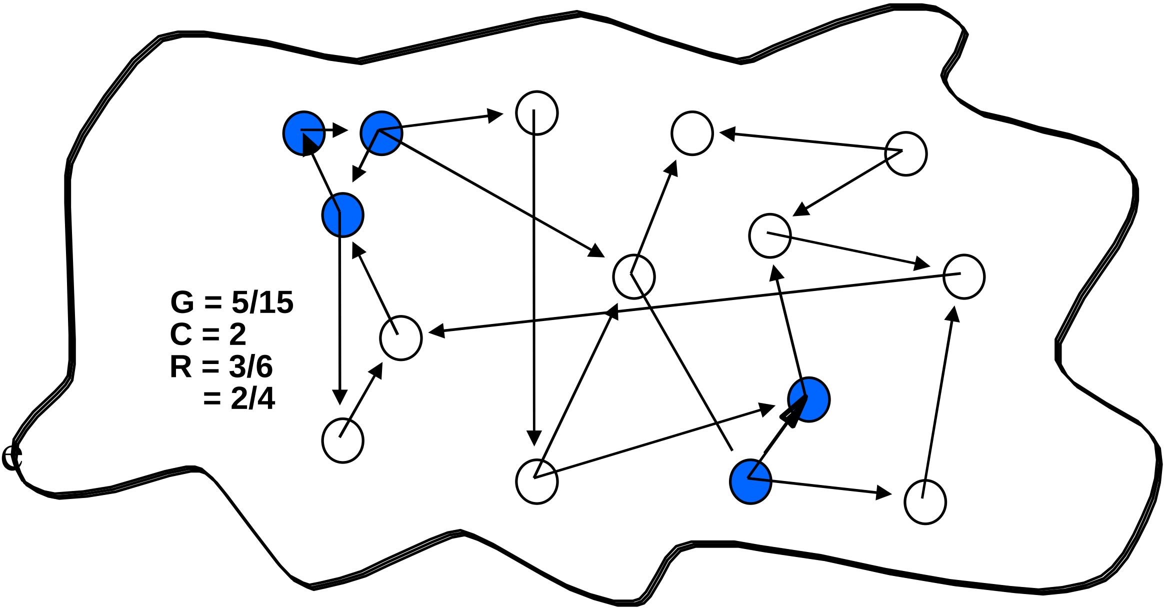
Quantifying topical locality

- Different ways to pose the question:
  - How quickly does semantic locality decay?
  - How fast is topic drift?
  - How quickly does content change as we surf away from a starting page?
- To answer these questions, let us consider exhaustive breadth-first crawls from 100 topic pages



The “link-cluster” conjecture

- Connection between semantic topology (relevance) and link topology (hypertext)
  - $G = \Pr[\mathrm{rel}(p)]$ ~ fraction of relevant/topical pages (topic generality)
  - $R = \Pr[\mathrm{rel}(p) \mid \mathrm{rel}(q) \text{ AND } \mathrm{link}(q,p)]$ ~ cond. prob. Given neighbor on topic
- Related nodes are clustered if ***R > G***
  - Necessary and sufficient condition for a random crawler to find pages related to start points
  - Example: 2 topical clusters with stronger modularity within each cluster than outside

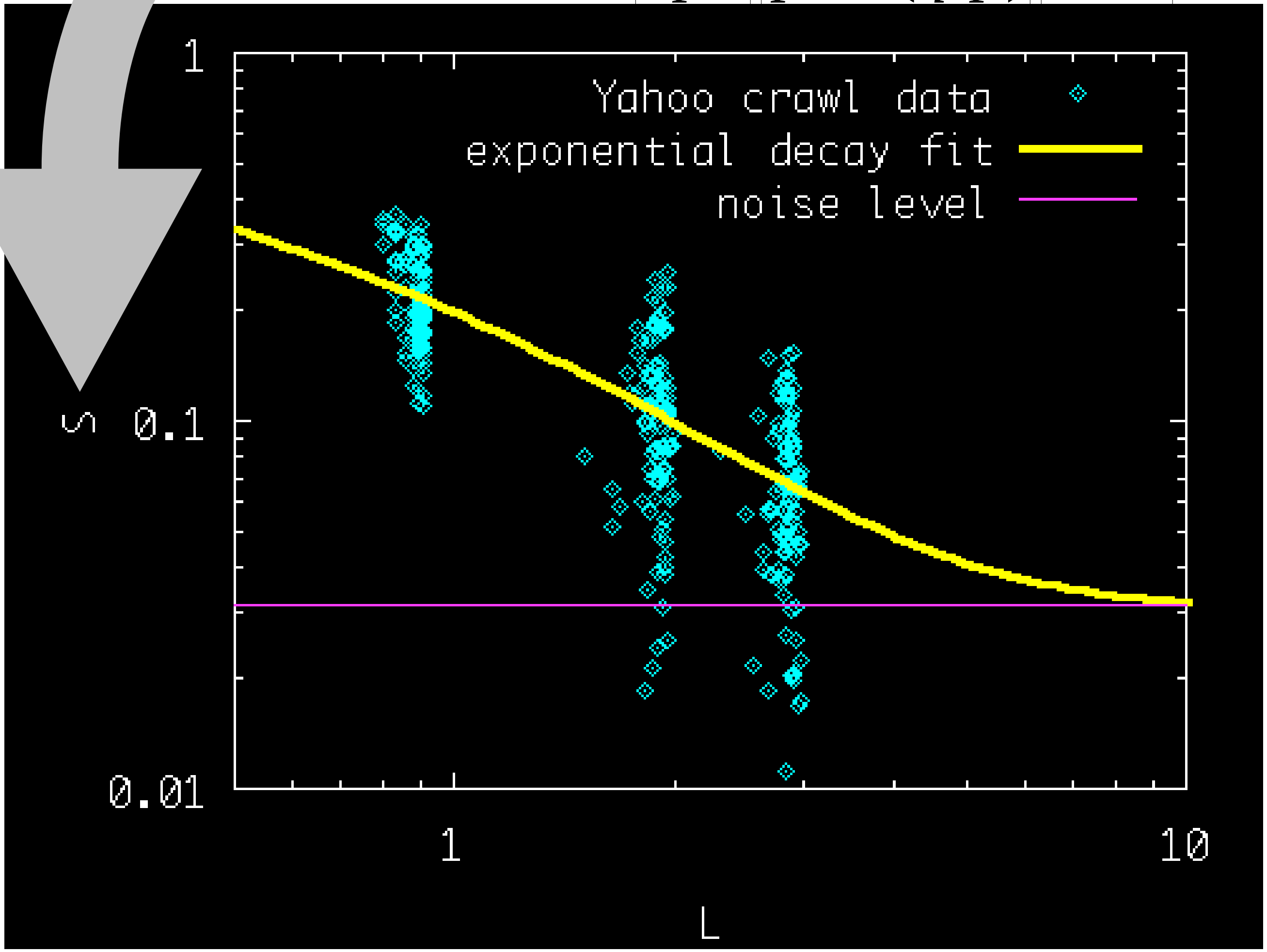


G = 5/15
C = 2
R = 3/6
= 2/4

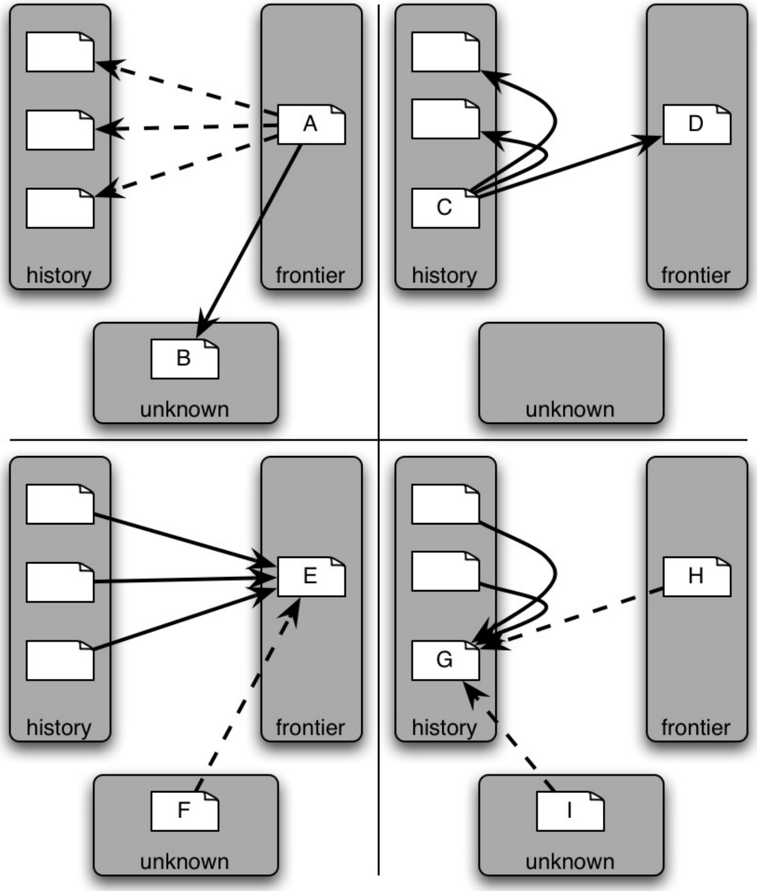
The “link-content” conjecture

$$S(q,d)\ ^{\text{o}}\ \frac{\underset{\{p:|path(q,p)|\text{£}d\}}{\text{å}} sim(q,p)}{|\{p:|path(q,p)|\text{£}\,d\}|}$$

- Correlation of lexical (content) and linkage topology
- **L(δ):** average link distance
- **S(δ):** average content similarity to start (topic) page from pages up to distance δ
- Correlation **ρ(L,S) = –0.76**



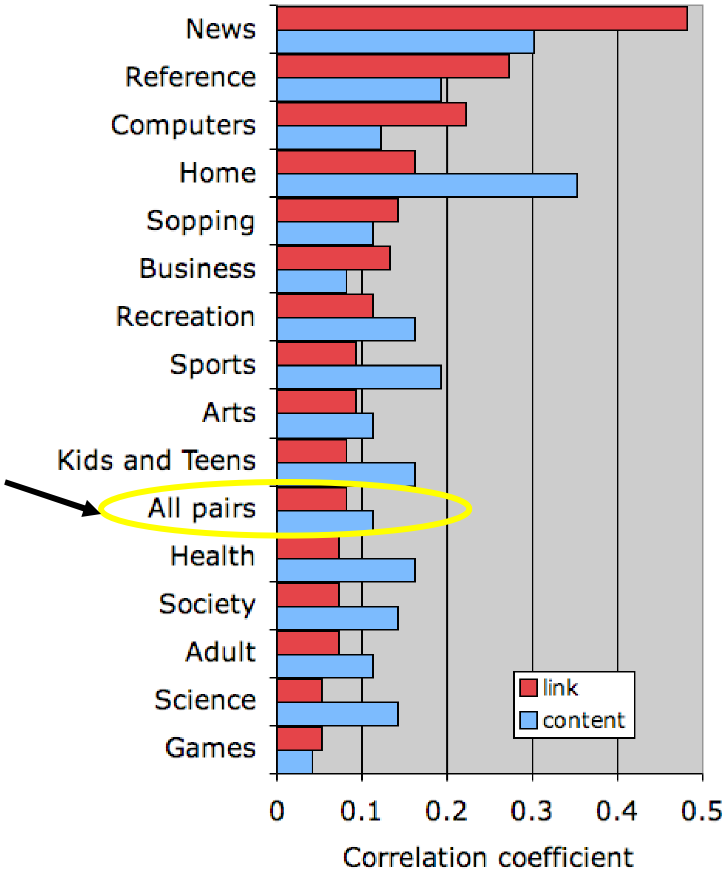
Topical locality-inspired tricks for topical crawlers



- Co-citation (a.k.a. sibling locality): A and C are good hubs, thus A and D should be given high priority
- Co-reference (a.k.a. blbliographic coupling): E and G are good authorities, thus E and H should be given high priority

Correlations between different similarity measures

- Semantic similarity measured from ODP, correlated with:
  - Content similarity: TF or TF-IDF vector cosine
  - Link similarity: Jaccard coefficient of (in+out) link neighborhoods
- Correlation overall is significant but weak
- Much stronger topical locality in some topics, e.g.:
  - Links very informative in news sources
  - Text very informative in recipes



Naïve Best-First

Simplest topical crawler: Frontier is priority queue based on text similarity between topic and parent page

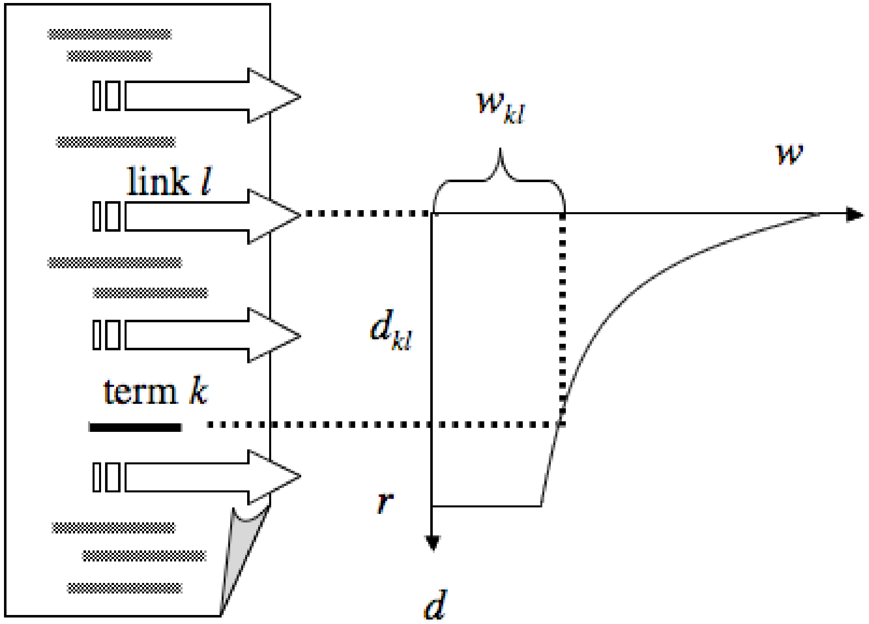
```
BestFirst(topic, seed_urls) {
   foreach link (seed_urls) {
      enqueue(frontier, link);
   }
   while (#frontier > 0 and visited < MAX_PAGES) {
      link := dequeue_link_with_max_score(frontier);
      doc := fetch_new_document(link);
      score :=  sim(topic, doc);
      foreach outlink (extract_links(doc)) {
         if (#frontier >= MAX_BUFFER) {
            dequeue_link_with_min_score(frontier);
         }
         enqueue(frontier, outlink, score);
      }
   }
}
```

Best-first variations

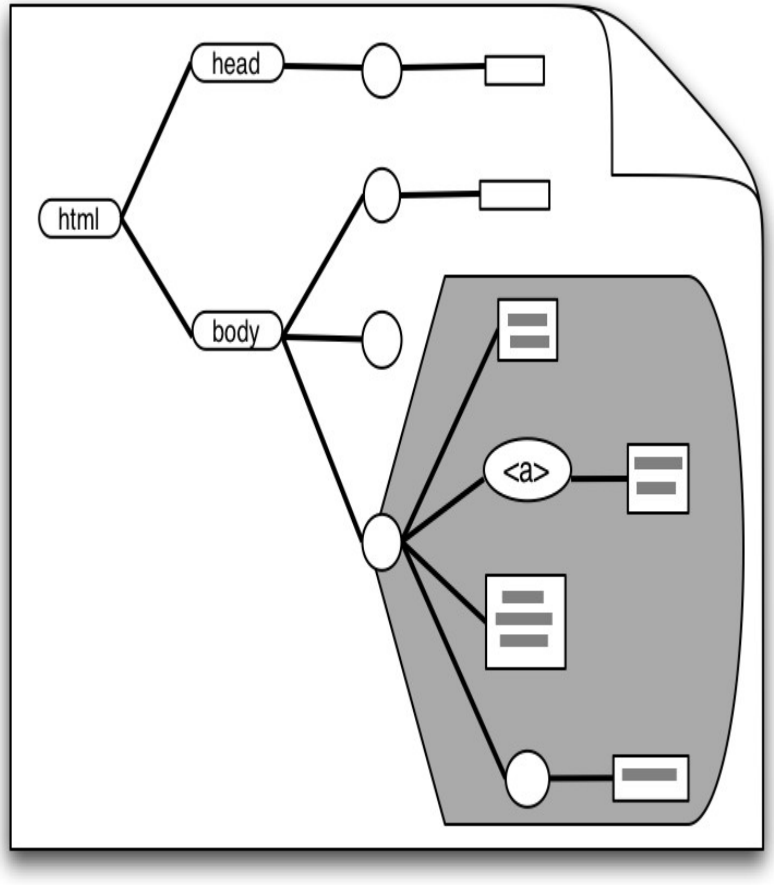
- Many in literature, mostly stemming from different ways to score unvisited URLs. E.g.:
  - Giving more importance to certain HTML markup in parent page
  - Extending text representation of parent page with anchor text from "grandparent" pages (SharkSearch)
  - Limiting link context to less than entire page
  - Exploiting topical locality (co-citation)
  - Exploration vs exploitation: relax priorities
- Any of these can be (and many have been) combined

Link context based on text neighborhood

- Often consider a fixed-size window, e.g. 50 words around anchor
- Can weigh links based on their distance from topic keywords within the document (InfoSpiders, Clever)
- Anchor text deserves extra importance

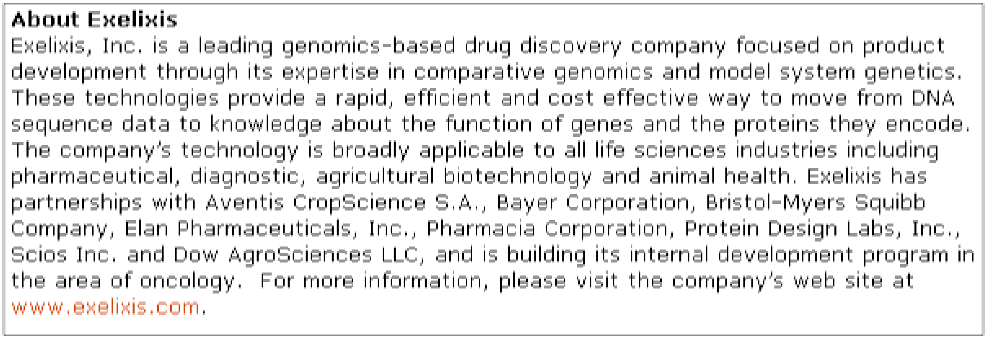


Link context based on DOM tree

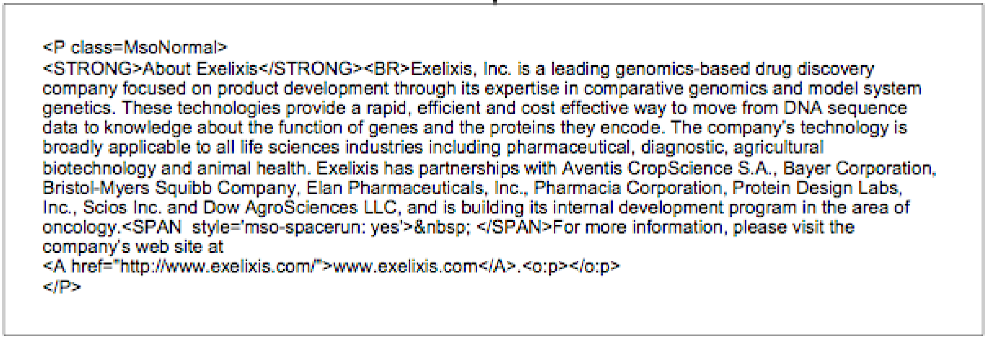


- Consider DOM subtree rooted at parent node of link’s <a> tag
- Or can go further up in the tree (Naïve Best-First is special case of entire document body)
- Trade-off between noise due to too small or too large context tree (Pant 2003)

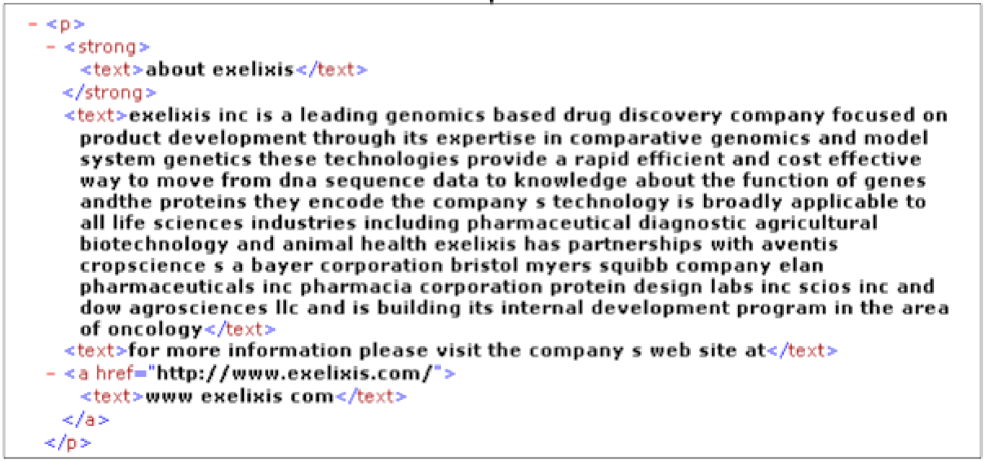
DOM context



About Exelixis
Exelixis, Inc. is a leading genomics-based drug discovery company focused on product development through its expertise in comparative genomics and model system genetics. These technologies provide a rapid, efficient and cost effective way to move from DNA sequence data to knowledge about the function of genes and the proteins they encode. The company's technology is broadly applicable to all life sciences industries including pharmaceutical, diagnostic, agricultural biotechnology and animal health. Exelixis has partnerships with Aventis CropScience S.A., Bayer Corporation, Bristol-Myers Squibb Company, Elan Pharmaceuticals, Inc., Pharmacia Corporation, Protein Design Labs, Inc., Scios Inc. and Dow AgroSciences LLC, and is building its internal development program in the area of oncology. For more information, please visit the company's web site at www.exelixis.com.

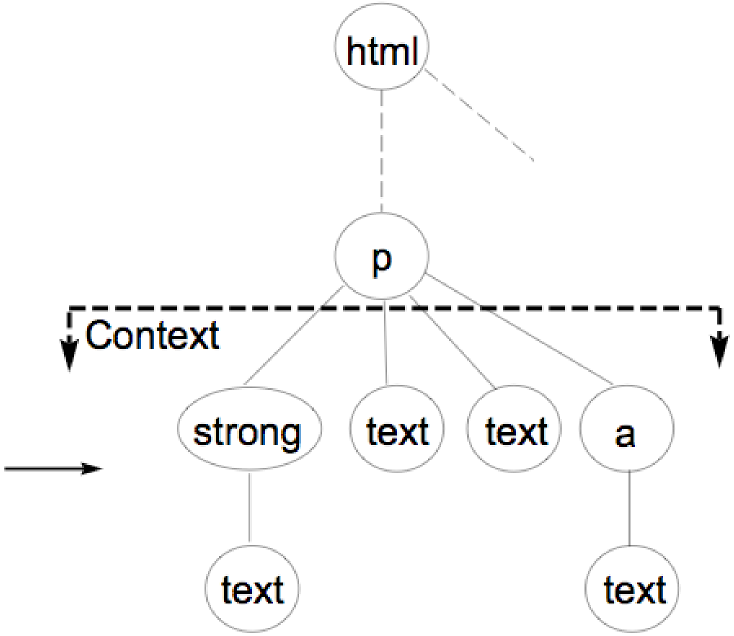


```
<P class=MsoNormal>
<STRONG>About Exelixis</STRONG><BR>Exelixis, Inc. is a leading genomics-based drug discovery
company focused on product development through its expertise in comparative genomics and model system
genetics. These technologies provide a rapid, efficient and cost effective way to move from DNA sequence
data to knowledge about the function of genes and the proteins they encode. The company's technology is
broadly applicable to all life sciences industries including pharmaceutical, diagnostic, agricultural
biotechnology and animal health. Exelixis has partnerships with Aventis CropScience S.A., Bayer Corporation,
Bristol-Myers Squibb Company, Elan Pharmaceuticals, Inc., Pharmacia Corporation, Protein Design Labs,
Inc., Scios Inc. and Dow AgroSciences LLC, and is building its internal development program in the area of
oncology.<SPAN  style='mso-spacerun: yes'>&nbsp; </SPAN>For more information, please visit the
company's web site at
<A href="http://www.exelixis.com/">www.exelixis.com</A>.<o:p></o:p>
</P>
```

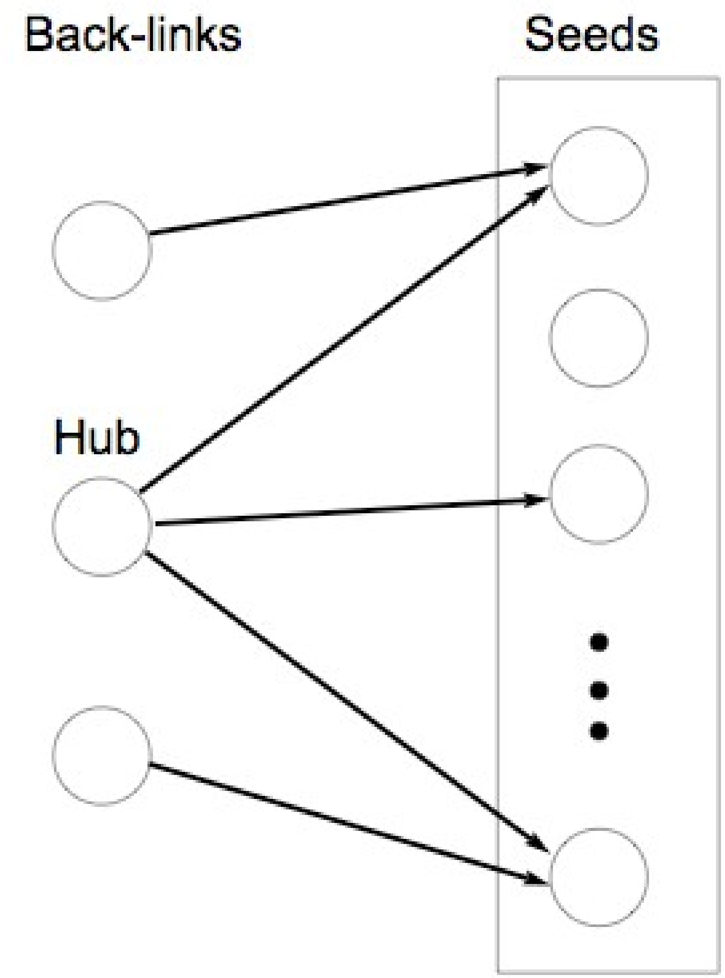


```
- <p>
  - <strong>
      <text>about exelixis</text>
    </strong>
    <text>exelixis inc is a leading genomics based drug discovery company focused on
      product development through its expertise in comparative genomics and model
      system genetics these technologies provide a rapid efficient and cost effective
      way to move from dna sequence data to knowledge about the function of genes
      andthe proteins they encode the company s technology is broadly applicable to
      all life sciences industries including pharmaceutical diagnostic agricultural
      biotechnology and animal health exelixis has partnerships with aventis
      cropscience s a bayer corporation bristol myers squibb company elan
      pharmaceuticals inc pharmacia corporation protein design labs inc scios inc and
      dow agrosciences llc and is building its internal development program in the area
      of oncology</text>
    <text>for more information please visit the company s web site at</text>
  - <a href="http://www.exelixis.com/">
      <text>www exelixis com</text>
    </a>
  </p>
```

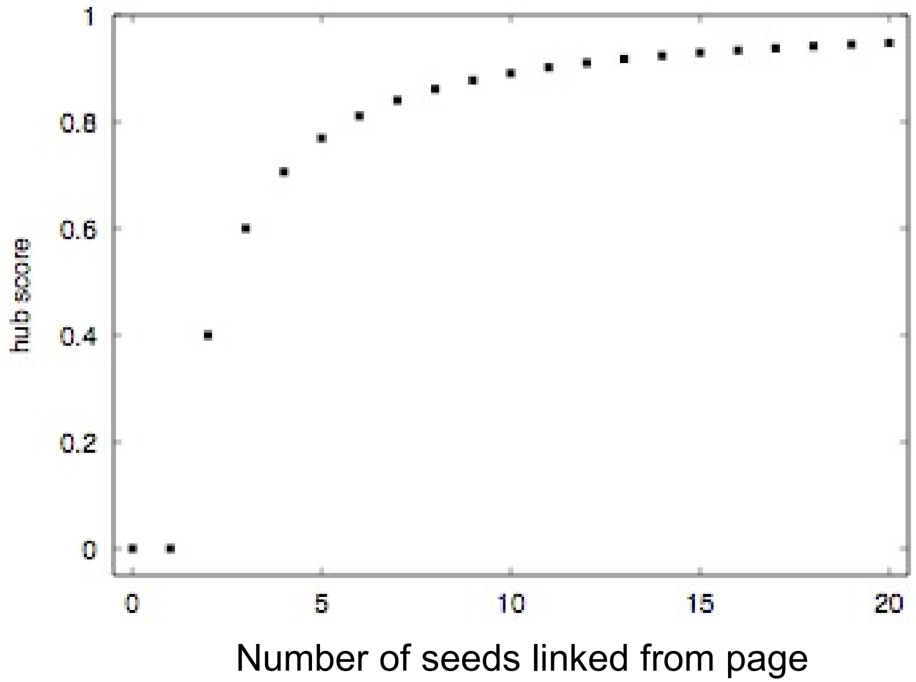
Link score = linear combination between page-based and context-based similarity score



Co-citation: hub scores

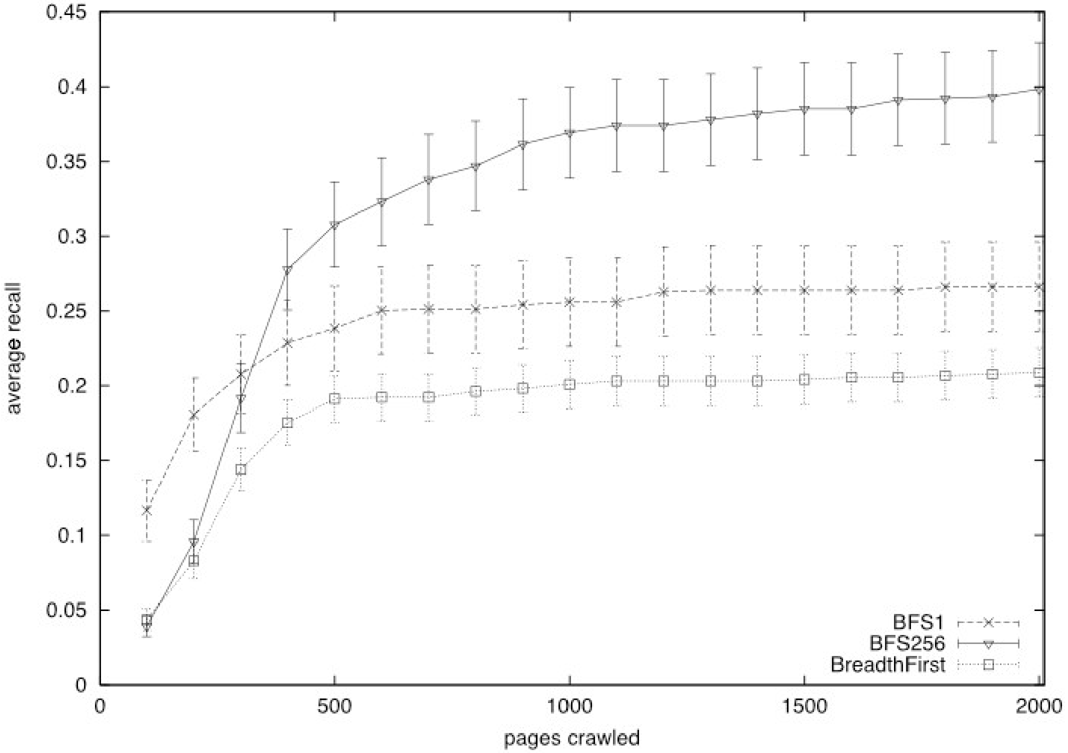


Link score$_{hub}$ = linear combination between link and hub score



Exploration vs Exploitation

- Best-*N*-First (or BFS*N*)
- Rather than re-sorting the frontier every time you add links, be lazy and sort only every *N* pages visited
- Empirically, being less greedy helps crawler performance significantly: escape “local topical traps” by exploring more



Pant et al. 2002

Outline

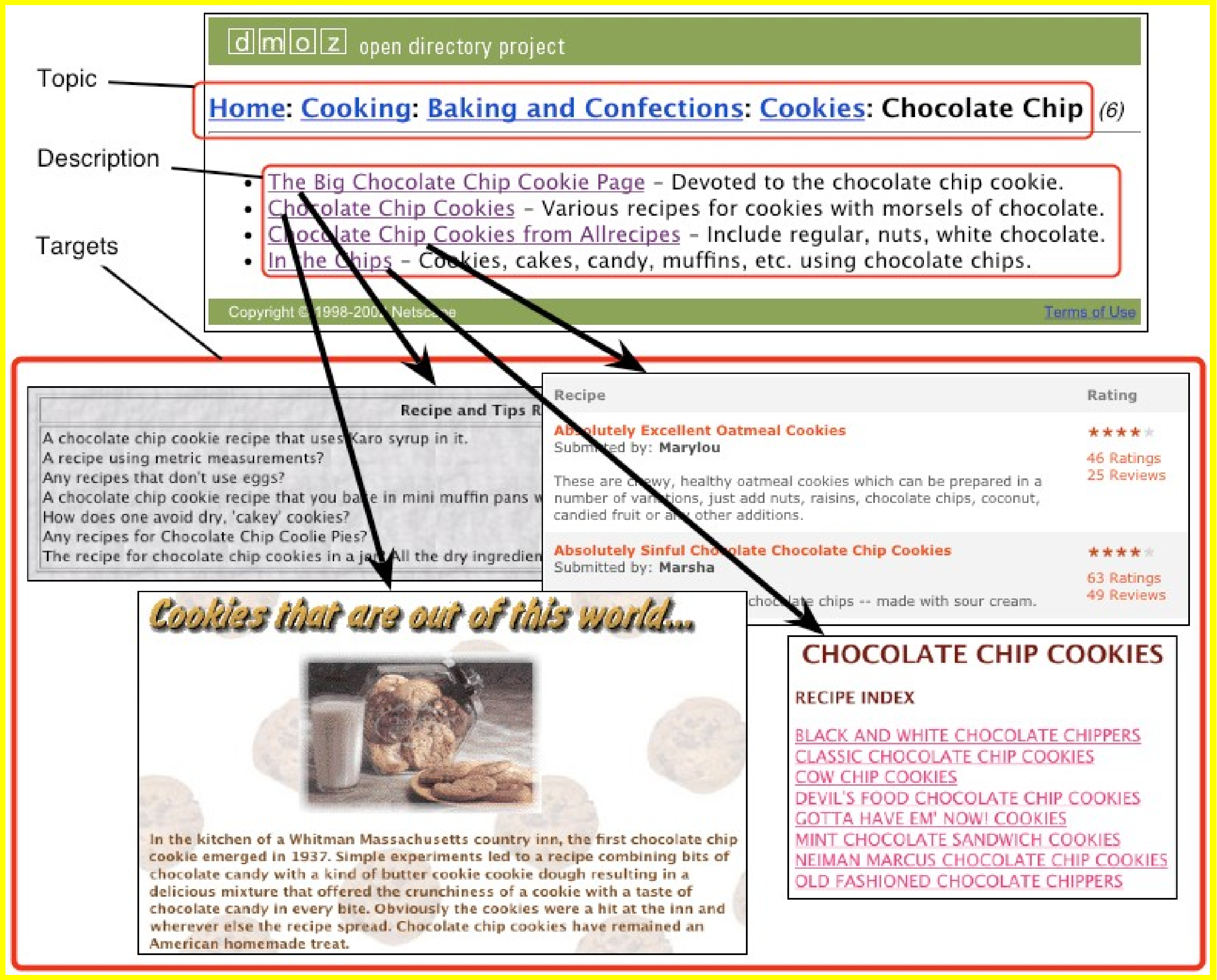
- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers

Evaluation of topical crawlers

- Goal: build “better” crawlers to support applications (Srinivasan & al. 2005)
- Build an unbiased evaluation framework
  - Define common tasks of measurable difficulty
  - Identify topics, relevant targets
  - Identify appropriate performance measures
    - Effectiveness: quality of crawler pages, order, etc.
    - Efficiency: separate CPU & memory of crawler algorithms from bandwidth & common utilities

Evaluation corpus = ODP + Web

- Automate evaluation using edited directories
- Different sources of relevance assessments



d m o z open directory project

Home: Cooking: Baking and Confections: Cookies: Chocolate Chip (6)

- The Big Chocolate Chip Cookie Page - Devoted to the chocolate chip cookie.
- Chocolate Chip Cookies - Various recipes for cookies with morsels of chocolate.
- Chocolate Chip Cookies from Allrecipes - Include regular, nuts, white chocolate.
- In the Chips - Cookies, cakes, candy, muffins, etc. using chocolate chips.

Copyright © 1998-2002 Netscape Terms of Use

Recipe and Tips R

A chocolate chip cookie recipe that uses Karo syrup in it.
A recipe using metric measurements?
Any recipes that don't use eggs?
A chocolate chip cookie recipe that you bake in mini muffin pans w
How does one avoid dry, 'cakey' cookies?
Any recipes for Chocolate Chip Coolie Pies?
The recipe for chocolate chip cookies in a jar. All the dry ingredien

Recipe	Rating
Absolutely Excellent Oatmeal Cookies Submitted by: **Marylou** These are chewy, healthy oatmeal cookies which can be prepared in a number of variations, just add nuts, raisins, chocolate chips, coconut, candied fruit or any other additions.	★★★★☆ 46 Ratings 25 Reviews
Absolutely Sinful Chocolate Chocolate Chip Cookies Submitted by: **Marsha** chocolate chips -- made with sour cream.	★★★★☆ 63 Ratings 49 Reviews

Cookies that are out of this world...

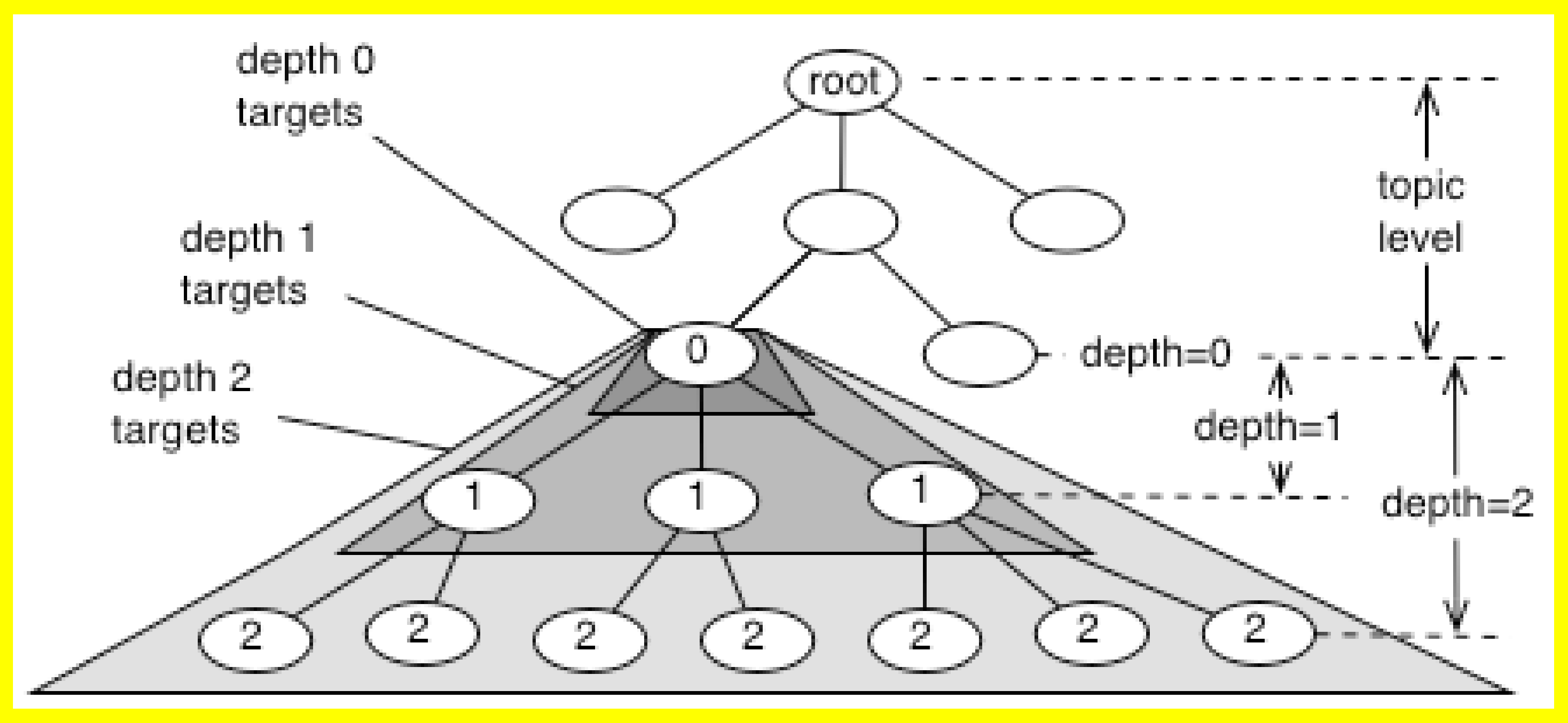
In the kitchen of a Whitman Massachusetts country inn, the first chocolate chip cookie emerged in 1937. Simple experiments led to a recipe combining bits of chocolate candy with a kind of butter cookie cookie dough resulting in a delicious mixture that offered the crunchiness of a cookie with a taste of chocolate candy in every bite. Obviously the cookies were a hit at the inn and wherever else the recipe spread. Chocolate chip cookies have remained an American homemade treat.

CHOCOLATE CHIP COOKIES

RECIPE INDEX

BLACK AND WHITE CHOCOLATE CHIPPERS
CLASSIC CHOCOLATE CHIP COOKIES
COW CHIP COOKIES
DEVIL'S FOOD CHOCOLATE CHIP COOKIES
GOTTA HAVE EM' NOW! COOKIES
MINT CHOCOLATE SANDWICH COOKIES
NEIMAN MARCUS CHOCOLATE CHIP COOKIES
OLD FASHIONED CHOCOLATE CHIPPERS

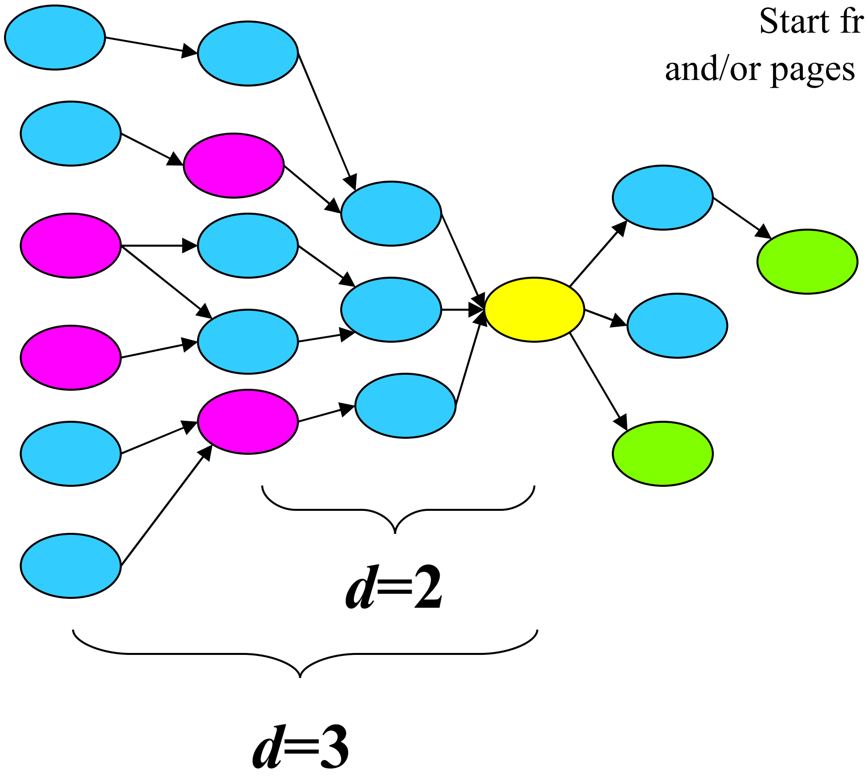
Topics and Targets



topic level ~ specificity
depth ~ generality

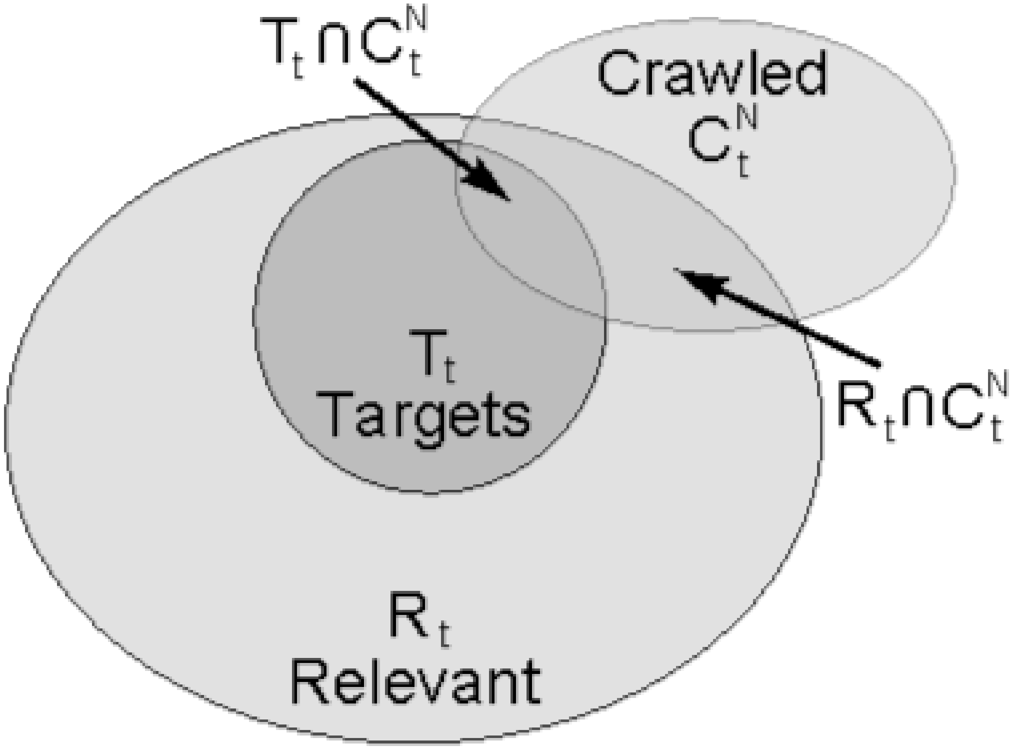
Tasks

Start from seeds, find targets
and/or pages similar to target descriptions



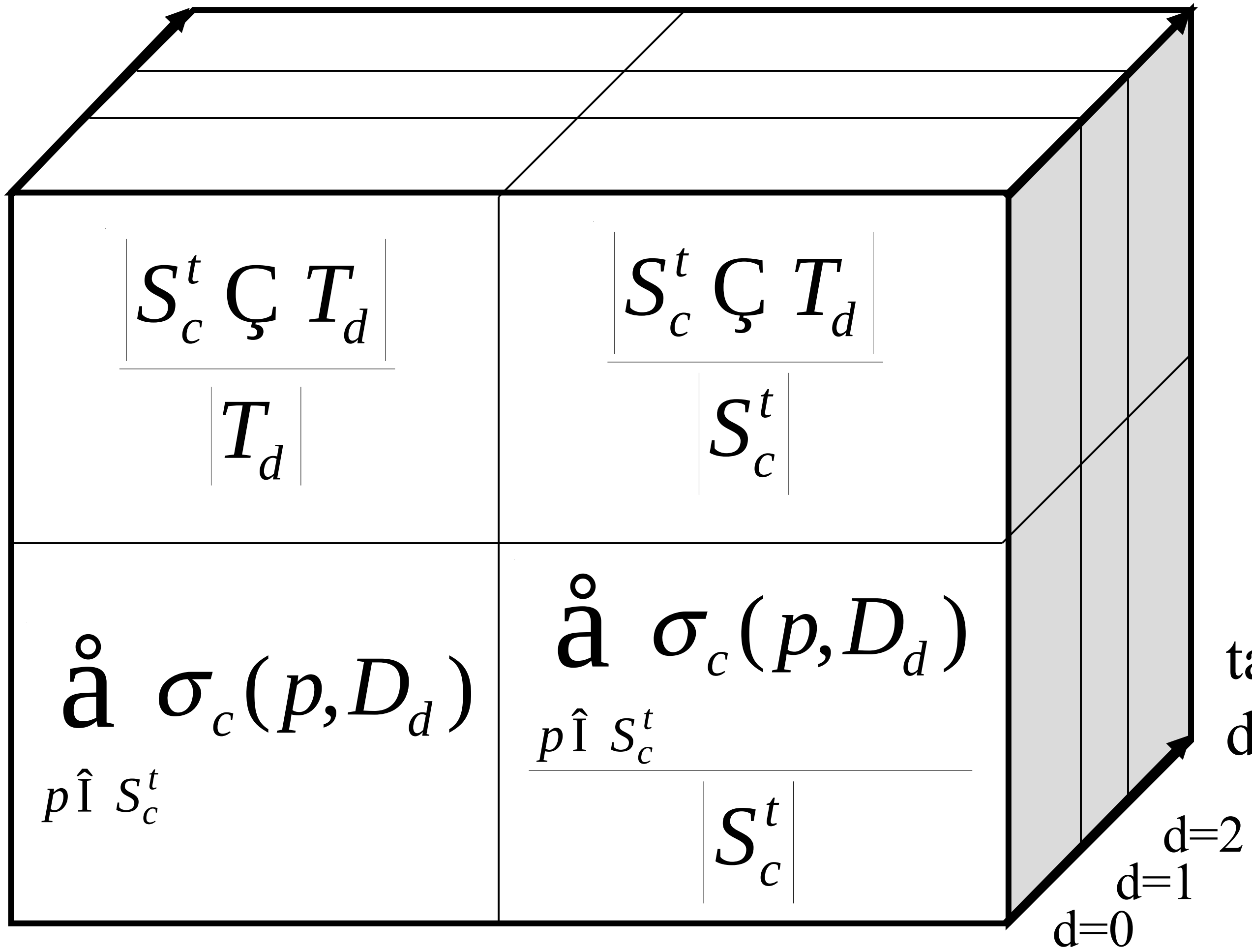
Back-crawl from targets to get seeds

Target based performance measures



Q: What assumption are we making? A: Independence!…

Performance matrix



	“recall”	“precision”
target pages	$\frac{\lvert S_c^t \text{ Ç } T_d \rvert}{\lvert T_d \rvert}$	$\frac{\lvert S_c^t \text{ Ç } T_d \rvert}{\lvert S_c^t \rvert}$
target descriptions	$\text{å}_{p \text{ Î } S_c^t} \sigma_c(p, D_d)$	$\frac{\text{å}_{p \text{ Î } S_c^t} \sigma_c(p, D_d)}{\lvert S_c^t \rvert}$

target depth

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments: social, collaborative, federated crawlers

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

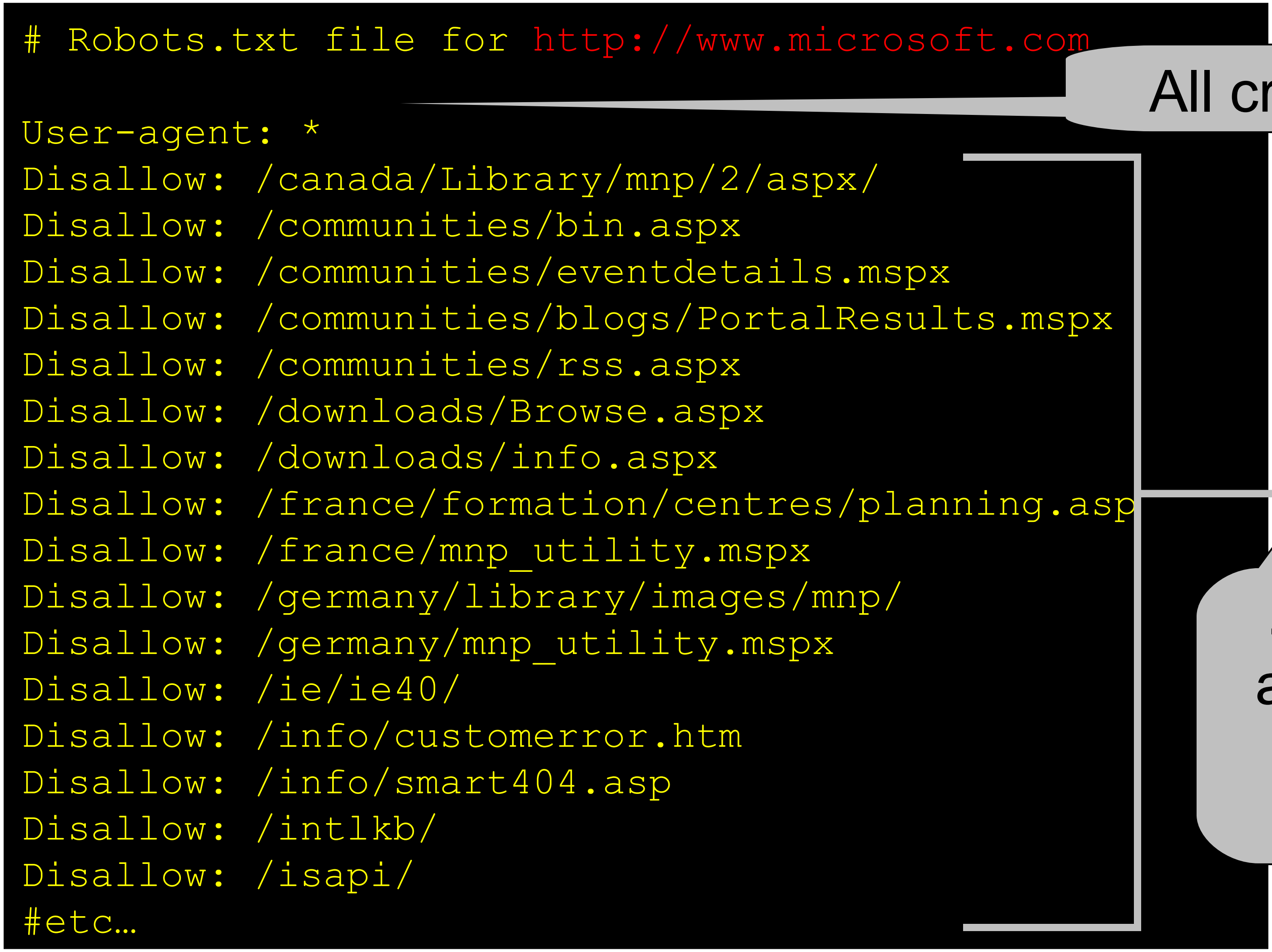
```
# robots.txt for http://www.apple.com/

User-agent: *
Disallow:
```

All crawlers…

…can go anywhere!

www.microsoft.com/robots.txt



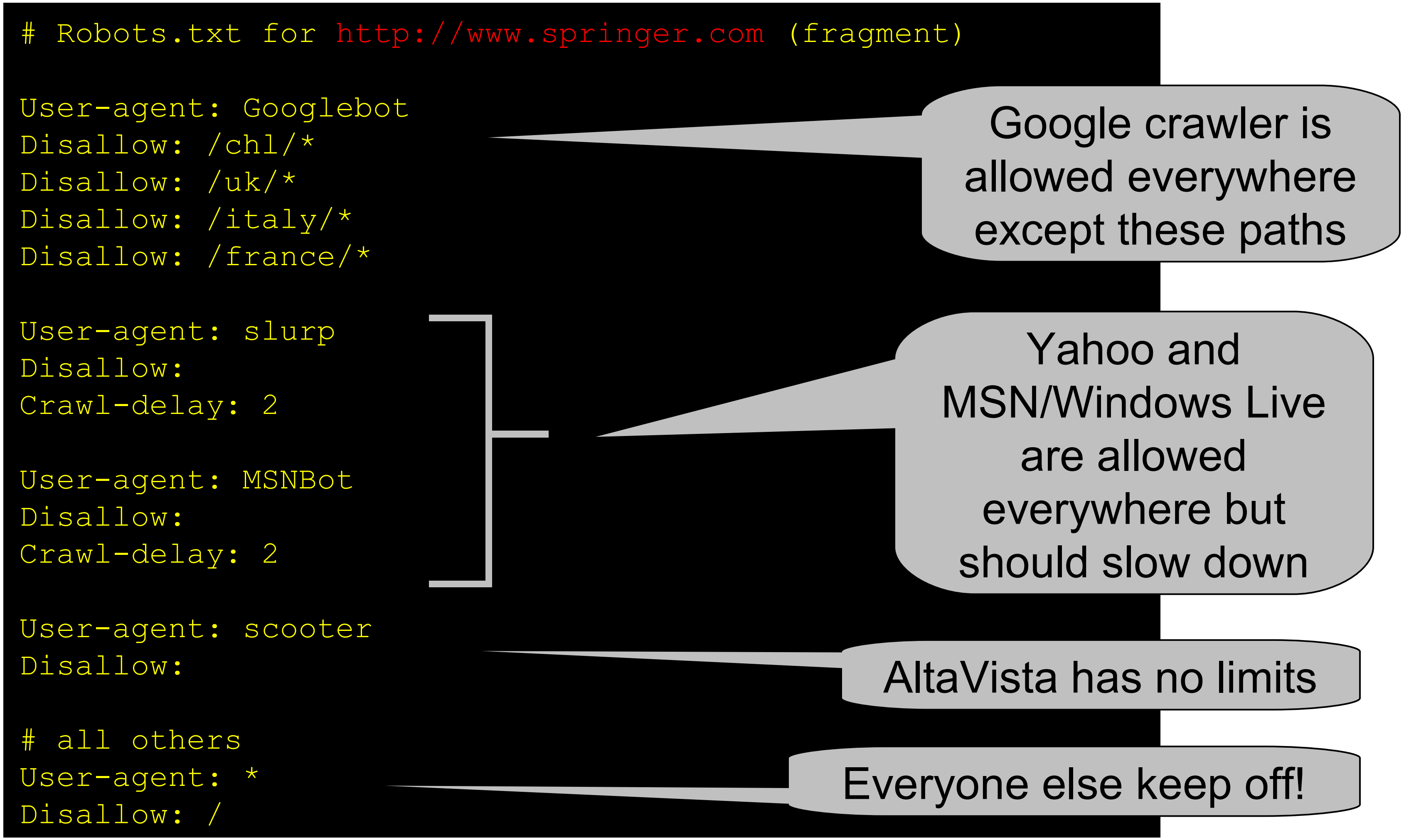
```
# Robots.txt file for http://www.microsoft.com

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?

Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers
- Evaluation of preferential crawlers
- Crawler ethics and conflicts
- New developments

New developments: social, collaborative, federated crawlers

- Idea: go beyond the "one-fits-all" model of centralized search engines
- Extend the search task to anyone, and distribute the crawling task
- Each search engine is a peer agent
- Agents collaborate by routing queries and results

Need crawling code?

- Reference C implementation of HTTP, HTML parsing, etc
  - w3c-libwww package from World-Wide Web Consortium: www.w3c.org/Library/
- LWP (Perl)
  - http://www.oreilly.com/catalog/perllwp/
  - http://search.cpan.org/~gaas/libwww-perl-5.804/
- Open source crawlers/search engines
  - Nutch: http://www.nutch.org/ (Jakarta Lucene: jakarta.apache.org/lucene/)
  - Heretrix: http://crawler.archive.org/
  - WIRE: http://www.cwr.cl/projects/WIRE/
  - Terrier: http://ir.dcs.gla.ac.uk/terrier/
- Open source topical crawlers, Best-First-N (Java)
  - http://informatics.indiana.edu/fil/IS/JavaCrawlers/
- Evaluation framework for topical crawlers (Perl)
  - http://informatics.indiana.edu/fil/IS/Framework/