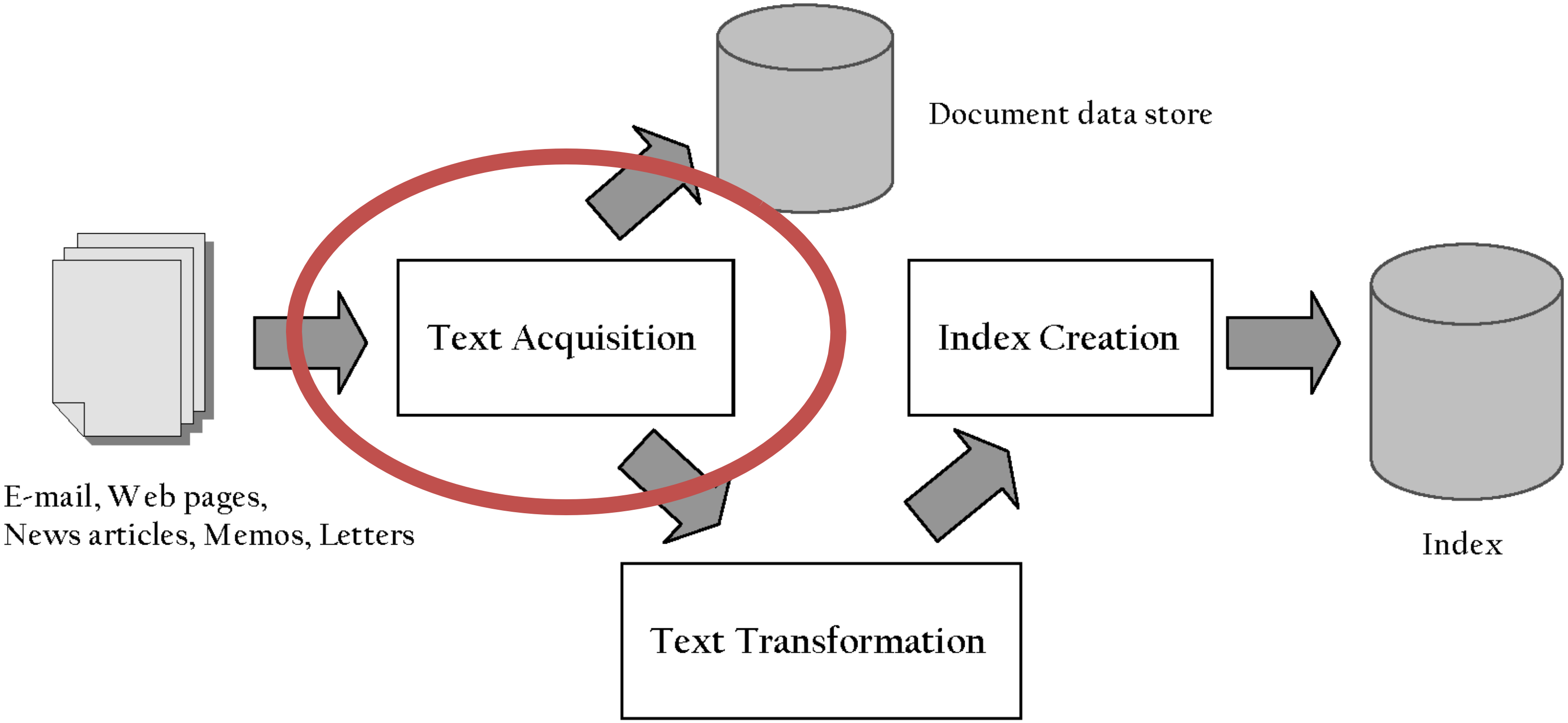
CS6200
Information Retrieval

Ghita Amor
ghitamor@ccs.neu.edu

Khoury College of Computer Sciences
Northeastern University

Indexing Process

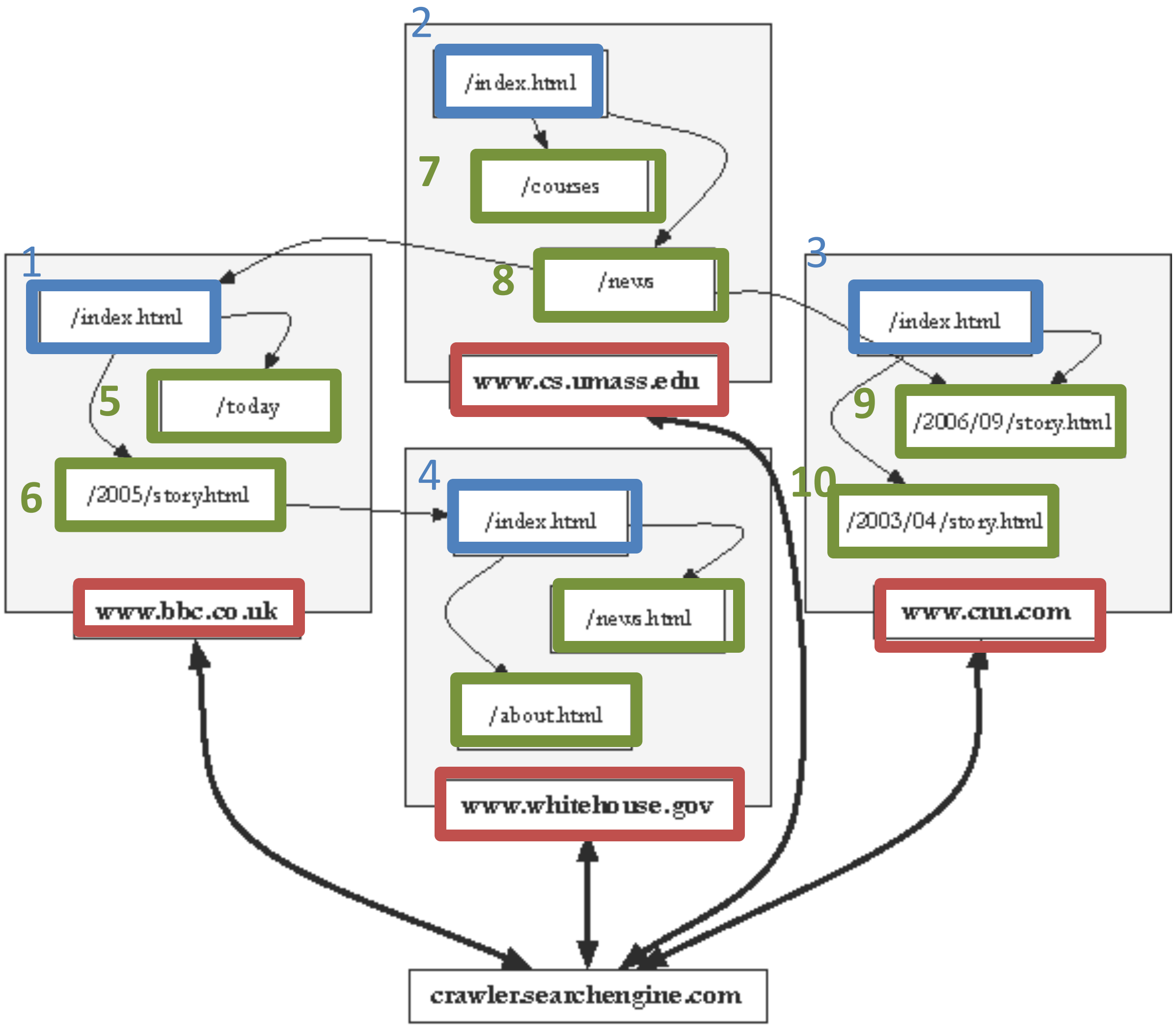


Basic Crawler

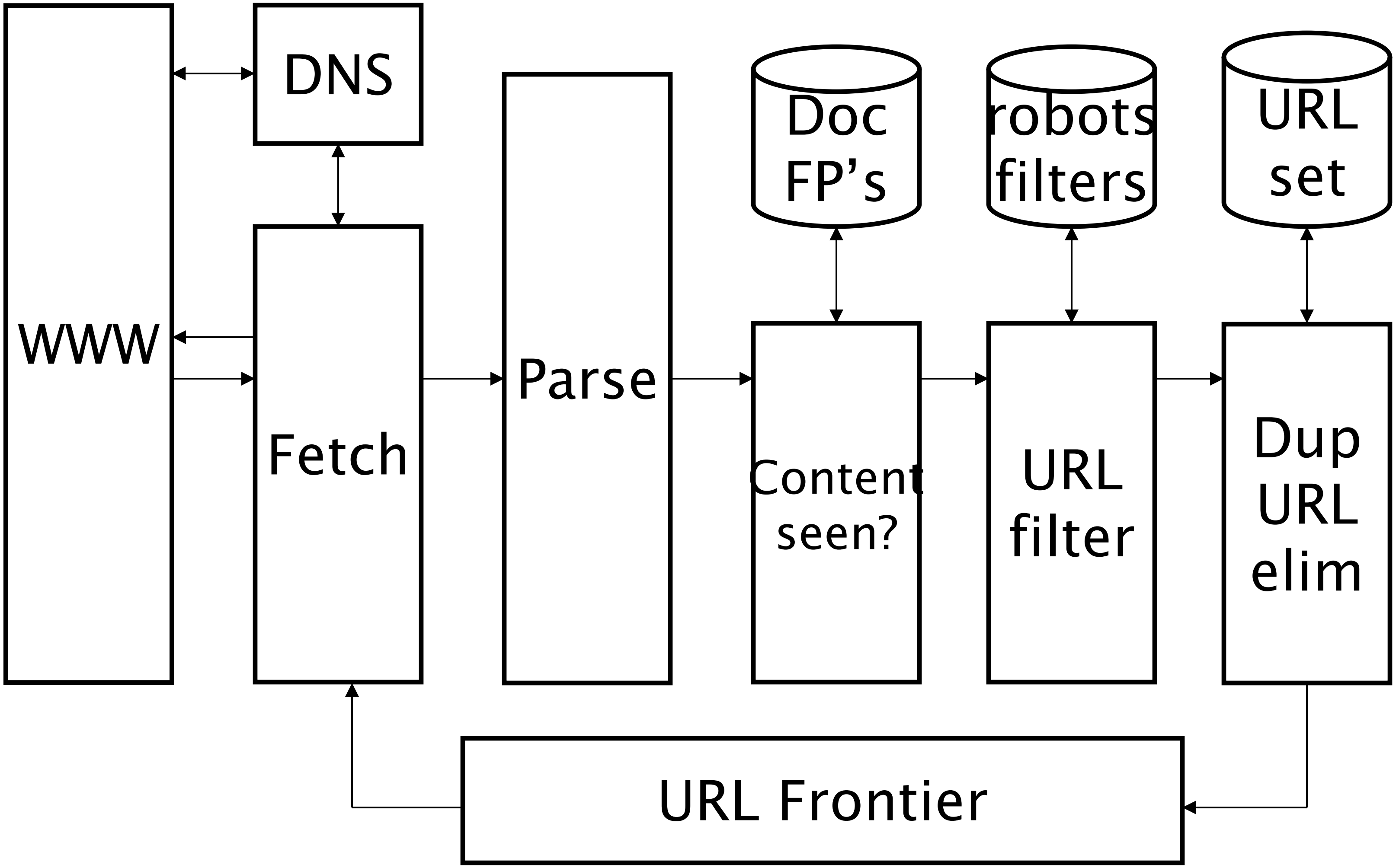
A crawler maintains a **frontier** – *a collection of pages to be crawled* – and iteratively selects and crawls pages from it.

1. The frontier is initialized with a list of seed pages.
2. Fetch and parse pages from the request queue
3. Find link tags that might contain other useful URLs to fetch
4. New URLs are processed and filtered before being added to the frontier.
5. Continue until no more new URLs or disk full

Web Crawler - Example



Basic Crawler Architecture



Topical Crawlers

Topical Crawlers focus on documents related to a particular topic of interest.

These crawlers are useful for improving the collection quality of general search engines, too.

Many search engines use a variety of topical crawlers to supplement their primary crawler.

A basic approach uses a topical set of seed URLs and text classifiers to decide whether links appear to be on topic.

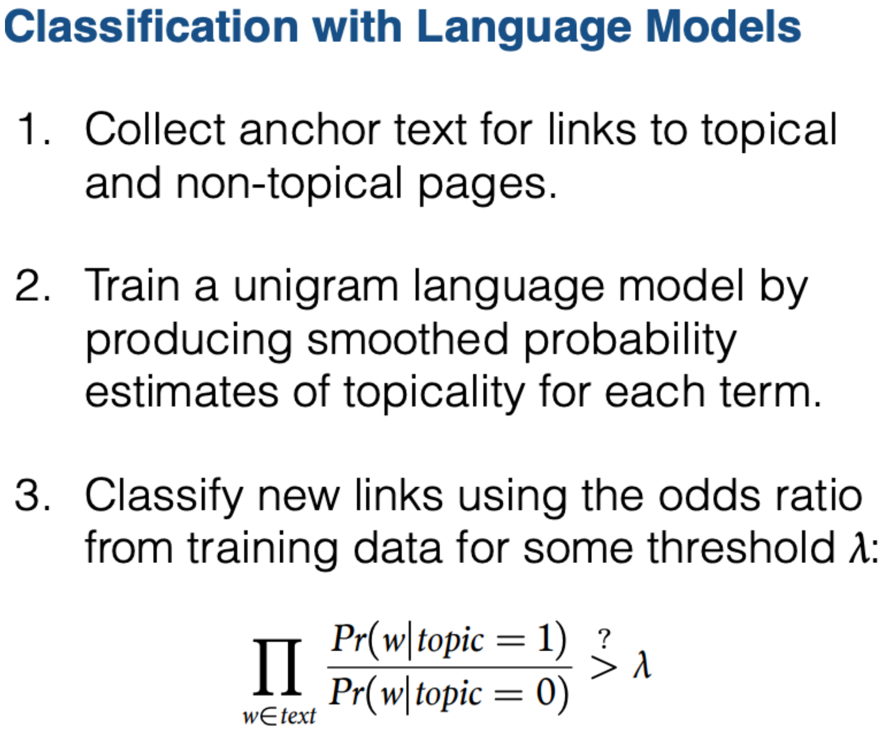
How do we know whether a page is on a given topic? We don't have access to content yet!

Text Classifiers

Text classification is a Machine Learning task that we'll cover later in the course.

IDEA for now: use properties of the URL, anchor text, and document to predict whether the URL links to a page on the topic of interest.

For example, we could use a unigram language model trained on anchor text for topical links.

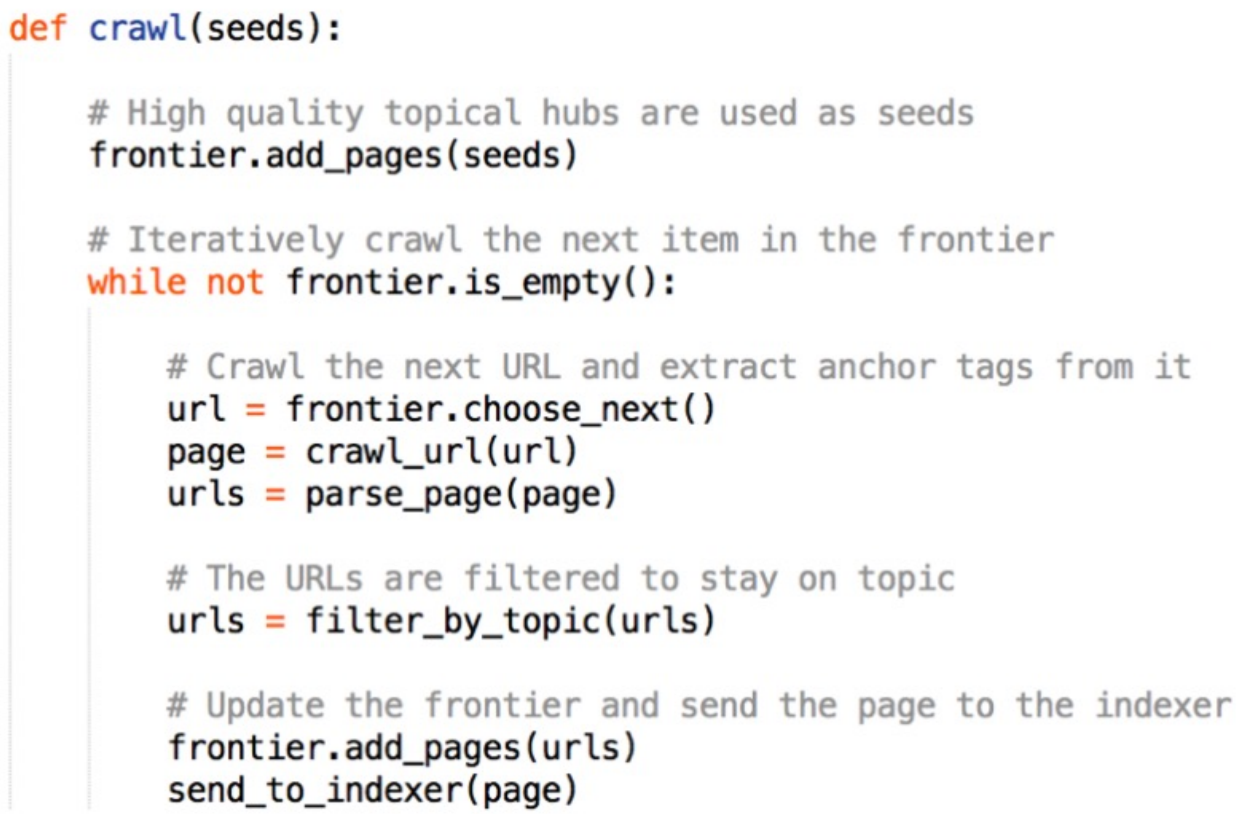


Classification with Language Models

1. Collect anchor text for links to topical and non-topical pages.
2. Train a unigram language model by producing smoothed probability estimates of topicality for each term.
3. Classify new links using the odds ratio from training data for some threshold λ:

$$\prod_{w \in text} \frac{Pr(w|topic = 1)}{Pr(w|topic = 0)} \overset{?}{>} \lambda$$

Topical Crawler



```python
def crawl(seeds):

    # High quality topical hubs are used as seeds
    frontier.add_pages(seeds)

    # Iteratively crawl the next item in the frontier
    while not frontier.is_empty():

        # Crawl the next URL and extract anchor tags from it
        url = frontier.choose_next()
        page = crawl_url(url)
        urls = parse_page(page)

        # The URLs are filtered to stay on topic
        urls = filter_by_topic(urls)

        # Update the frontier and send the page to the indexer
        frontier.add_pages(urls)
        send_to_indexer(page)
```

Basic Topical Crawler

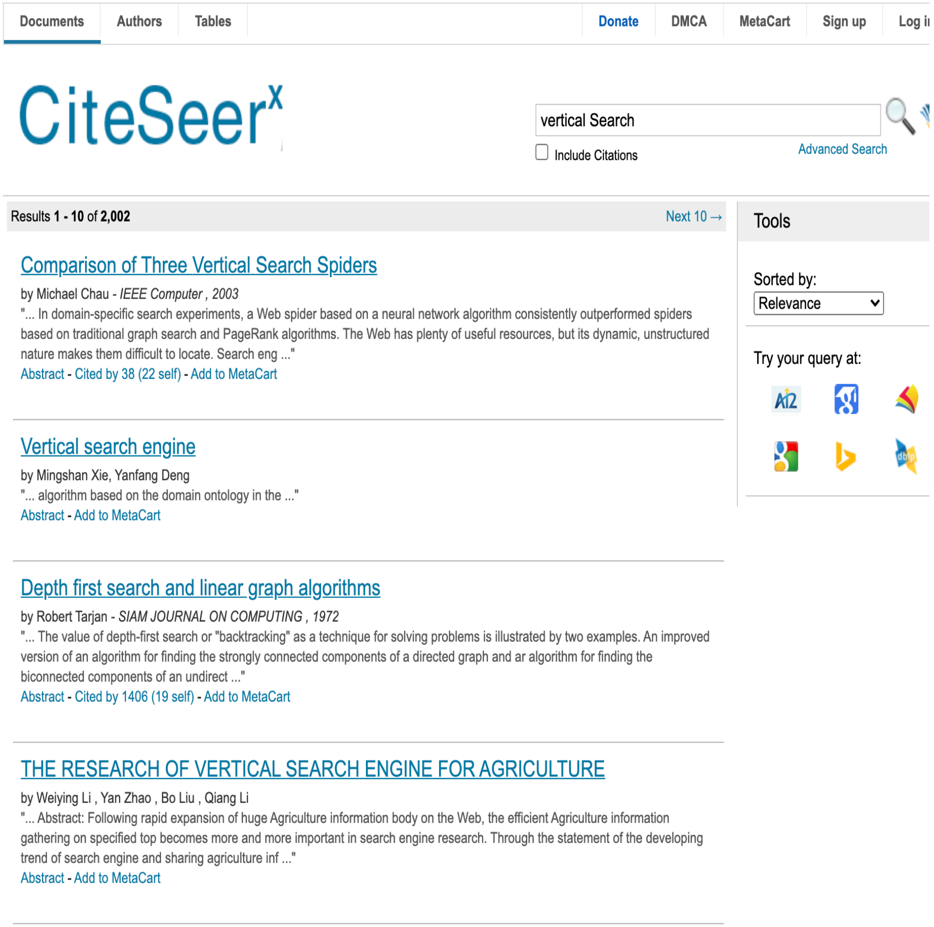
How Does all of this Relate to HW3?

Vertical Search

Vertical Search engines focus on a particular domain of information.

The primary **difference** between vertical and general search engines is the set of documents they crawl.

Vertical Search engines typically use what are known as topical crawlers.



Results **1 - 10** of **2,002** Next 10 →

Comparison of Three Vertical Search Spiders

by Michael Chau - *IEEE Computer*, *2003*

"... In domain-specific search experiments, a Web spider based on a neural network algorithm consistently outperformed spiders based on traditional graph search and PageRank algorithms. The Web has plenty of useful resources, but its dynamic, unstructured nature makes them difficult to locate. Search eng ..."

Abstract - Cited by 38 (22 self) - Add to MetaCart

Vertical search engine

by Mingshan Xie, Yanfang Deng

"... algorithm based on the domain ontology in the ..."

Abstract - Add to MetaCart

Depth first search and linear graph algorithms

by Robert Tarjan - *SIAM JOURNAL ON COMPUTING*, *1972*

"... The value of depth-first search or "backtracking" as a technique for solving problems is illustrated by two examples. An improved version of an algorithm for finding the strongly connected components of a directed graph and ar algorithm for finding the biconnected components of an undirect ..."

Abstract - Cited by 1406 (19 self) - Add to MetaCart

THE RESEARCH OF VERTICAL SEARCH ENGINE FOR AGRICULTURE

by Weiying Li , Yan Zhao , Bo Liu , Qiang Li

"... Abstract: Following rapid expansion of huge Agriculture information body on the Web, the efficient Agriculture information gathering on specified top becomes more and more important in search engine research. Through the statement of the developing trend of search engine and sharing agriculture inf ..."

Abstract - Add to MetaCart

Web Crawler - Coverage

The first goal of an Internet crawler is to provide adequate coverage. **Coverage** is the fraction of available content you've crawled.

Challenges include:

- Discovering new pages and web sites as they appear online.
- Duplicate site detection, so you don't waste time re-crawling content you already have.
- Avoiding **spider traps** – configurations of links that would cause a naive crawler to make an infinite series of requests

Coverage

The Internet is too large and changes too rapidly for any crawler to be able to crawl and index it all. Instead, a crawler should focus on **strategic crawling** to balance coverage and freshness.

A crawler should **prioritize crawling high-quality content** to better answer user queries. The Internet contains a lot of spam, redundant information, and pages which aren't likely to be relevant to users' information needs

Good coverage is obtained by carefully selecting seed URLs and using a good page selection policy to decide what to crawl next.

Breadth-first search is adequate when you have simple needs, but many techniques outperform it. It particularly helps to have an existing index from a previous crawl.

Deep Web

A substantial fraction of web pages remains uncrawled and unindexed by search engines. These pages are known as the **deep web**.

Three broad categories:

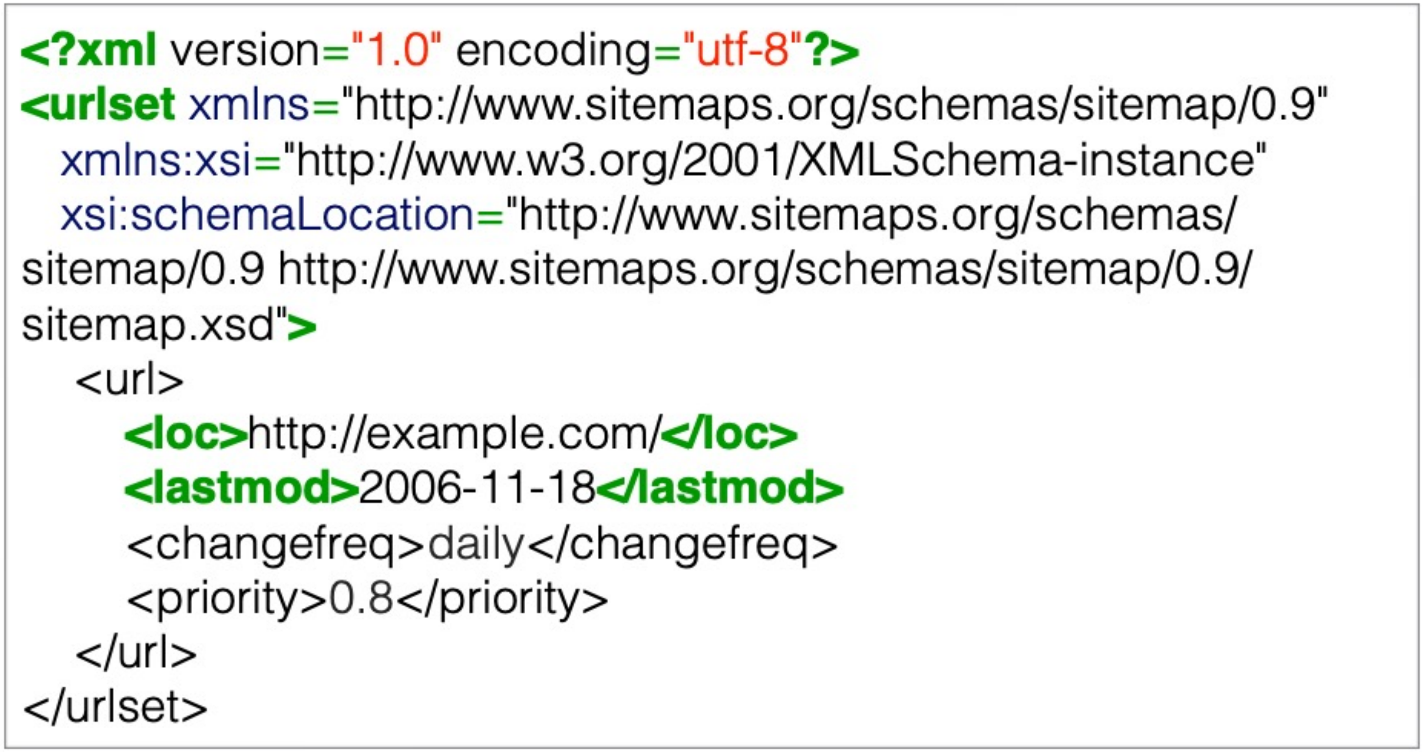
- private web sites either protected by passwords or have no incoming links
- Dynamically generated pages, that use JavaScript, Flash, or another client-side language to generate links
- Some pages are intentionally hidden, using `robots.txt` or more sophisticated approaches such as “darknet” software.

Sitemaps

Sitemaps contain lists of URLs and data about those URLs, such as modification time and modification frequency

- Generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check a page for changes

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>
```

Example courtesy Wikipedia

Web Crawler

Web crawlers are generally **distributed** and **multithreaded**

They are capable of taxing the capabilities of most web servers - This is particularly true of the crawlers for major businesses, such as Bing and Google.

Also, some content should not be crawled at all

- Some web content is considered private or under copyright, and its owners prefer that it not be crawled and indexed.
- Other URLs implement API calls and don't lead to indexable content.

Web Crawler - Politeness

Crawling the web consumes resources on the servers we're visiting. **Politeness** is a set of policies a well-behaved crawler should obey in order to be respectful of those resources.

Politeness policies:

- Requests to the same domain should be made with a reasonable delay.
- The total bandwidth consumed from a single site should be limited.
- Site owners' preferences, expressed by files such as `robots.txt`, should be respected.

Request Intervals

Limit the rate of requests your crawler makes to the same domain. Too-frequent requests are the main way a crawler can harm a web site.

Typical crawler delays are in the range of 10-15 seconds per request. You should never crawl more than one page per second from the same domain. For large sites, it can take days or weeks to crawl the entire domain – this is preferable to overloading their site

If the site's `robots.txt` file has a Crawl-delay directive, it should be honored!

In a **distributed crawler**, all requests for the same domain are typically sent to the same crawler instance to easily throttle the rate of requests.

Controlling Crawling

Web site administrators can express their crawling preferences by hosting a page at `/robots.txt`. Every crawler should honor these preferences.

```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public/

User-agent: FavoredCrawler
Disallow:

Sitemap: http://mysite.com/sitemap.xml.gz
```

Visit ***http://www.robotstxt.org/robotstxt.html*** *for details on the protocol*

Web Crawler - Freshness

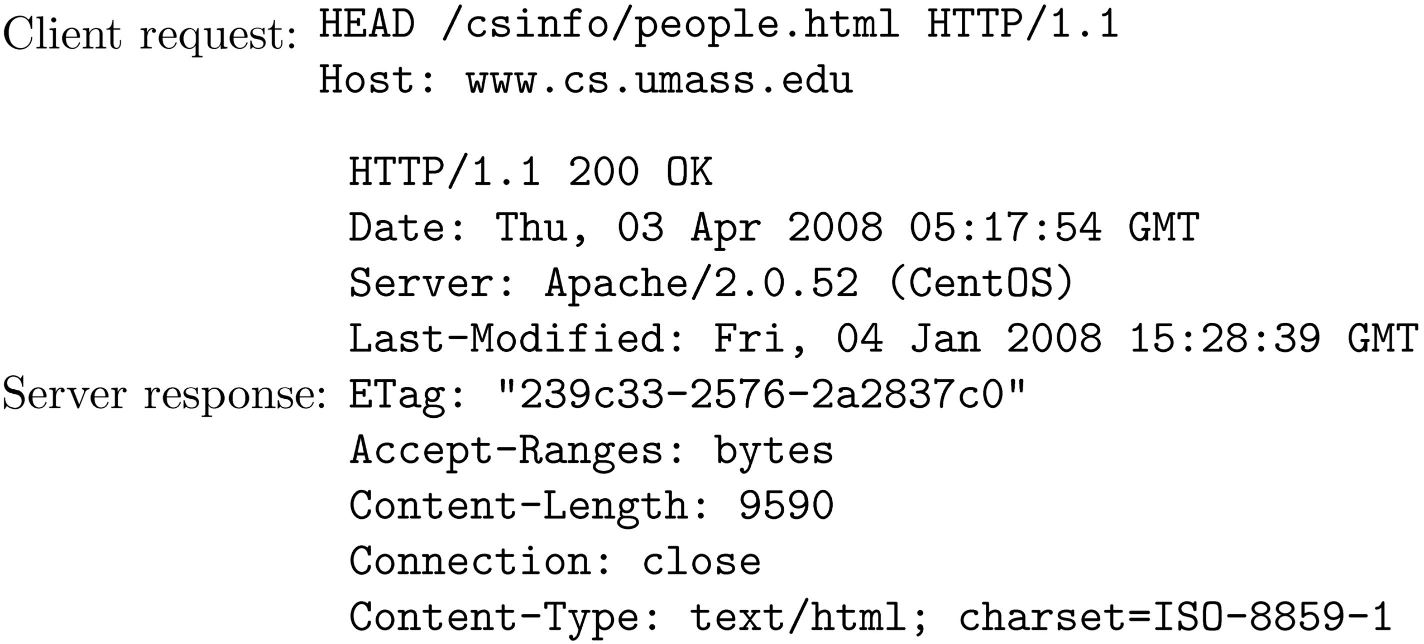
Coverage is often at odds with freshness. **Freshness** is the recency of the content in your index. If a page you've already crawled changes, you'd like to re-index it.

Challenges include:

- Making sure your search engine provides good results for breaking news.
- Identifying the pages or sites which tend to be updated often.
- Balancing your limited crawling resources between new sites (coverage) and updated sites (freshness).

Freshness

A crawler can determine whether a page has changed by making an **HTTP HEAD request** - returns information about page, not the page itself.



Client request:

```
HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
```

Server response:

```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Freshness

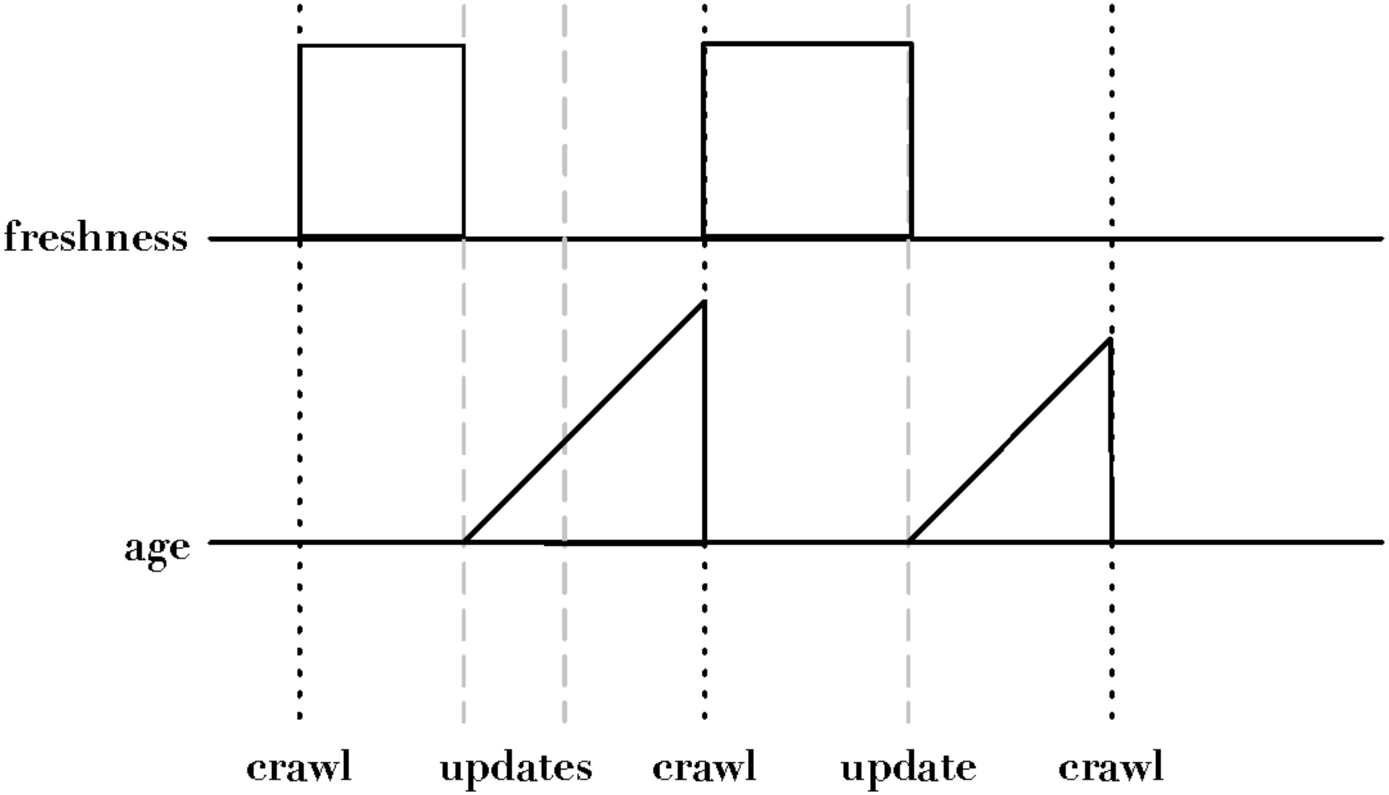
Freshness is the proportion of pages that are fresh

Not possible to constantly check all pages - must check important pages and pages that change frequently

Optimizing for this metric is a poor strategy: it can lead the crawler to ignore important sites.

Instead, it's better to re-crawl pages when the age of the last crawled version exceeds some limit. The age of a page is the elapsed time since the first update after the most recent crawl.

Freshness vs. Age



Freshness is binary, age is continuous.

Expected Age

Expected age of a page, *t* days after it was last crawled depends on its update probability:

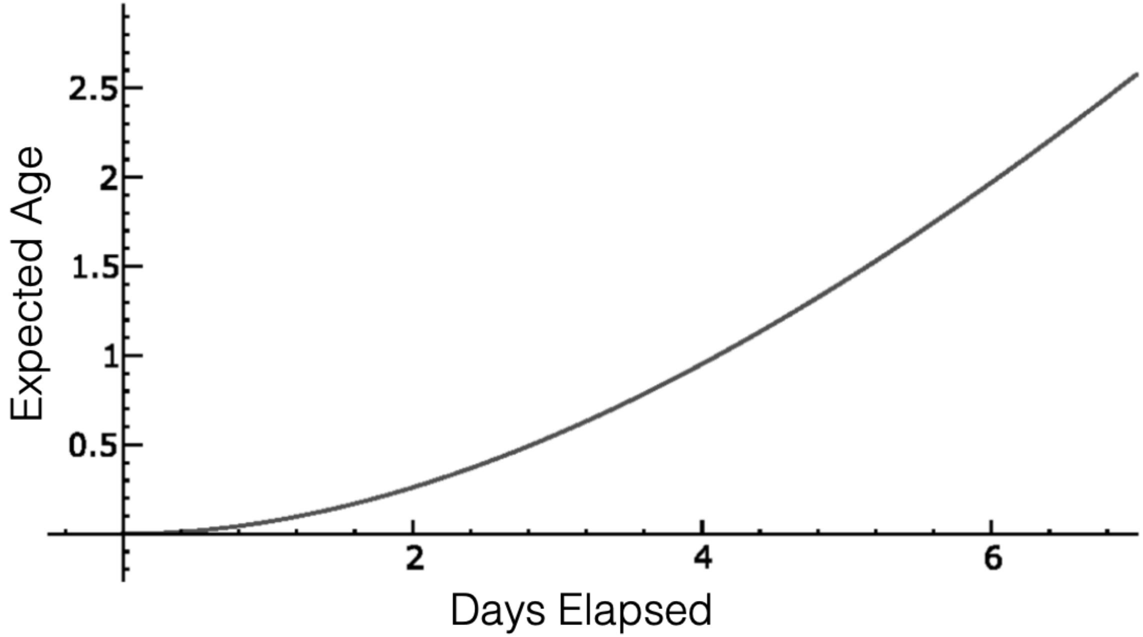
$$\text{Age}(\lambda, t) = \int_0^t P(\text{page changed at time } x)(t - x)dx$$

Web page updates follow the Poisson distribution on average – the time until the next update is governed by an exponential distribution

$$\text{Age}(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx$$

Cost of Not Re-crawling

The cost of not re-crawling a page grows exponentially in the time since the last crawl. E.g. Days Elapsed Days Elapsed, with page update frequency λ = 1/7 days:



Freshness vs. Coverage

The opposing needs of Freshness and Coverage need to be balanced in the scoring function used to select the next page to crawl.

Finding an optimal balance is still an open question. Fairly recent studies have shown that even large name-brand search engines only do a modest job at finding the most recent content.

However, a reasonable approach is to include a term in the page priority function for the expected age of the page content. For important domains, you can track the site-wide update frequency λ

And more …

Aside from these concerns, **a good crawler should:**

- Focus on crawling high-quality web sites.
- Be distributed and scalable, and make efficient use of server resources.
- Crawl web sites from a geographically-close data center (when possible).
- Be extensible, so it can handle different protocols and web content types appropriately.

Pitfalls of Crawling

A **breadth-first search implementation** of crawling is not sufficient for coverage, freshness, spam avoidance, or other needs of a real crawler.

Scaling the crawler up takes careful engineering, and often detailed systems knowledge of the hardware architecture you’re developing for.

Goals of Crawling

A good crawler will balance several factors:

Coverage: Pages should be selected to maximize the number of distinct high-quality pages.

Freshness: Recrawl pages with a frequency that approximates the rate of change of that page.

Performance: Each machine should crawl thousands of pages per second.

Politeness: Successive requests to the same domain are not frequent, and the site owners’ crawler policies are respected.

Distributed: Execute across multiple machines