

Crawls & Feeds

406.424 Internet Applications

Jonghun Park

jonghun@snu.ac.kr

**Dept. of Industrial Eng.
Seoul National University**

9/1/2010



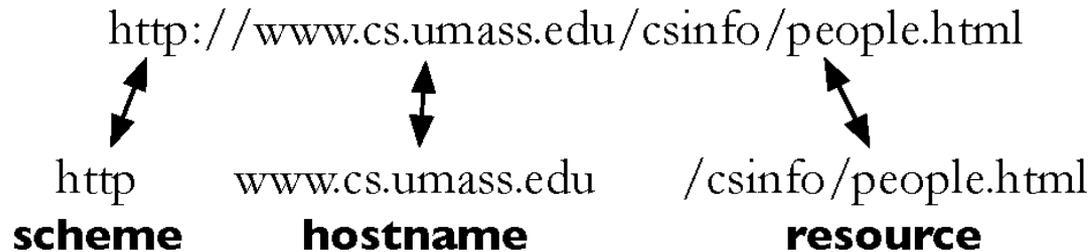
web crawler

- finds and downloads web pages automatically
- web is huge and constantly growing
 - at least tens of billions of pages
- web is not under the control of search engine providers
- web pages are constantly changing



retrieving web pages

- every page has a unique **uniform resource locator** (URL)
- web pages are stored on web servers that use HTTP to exchange information with client software
- e.g.,

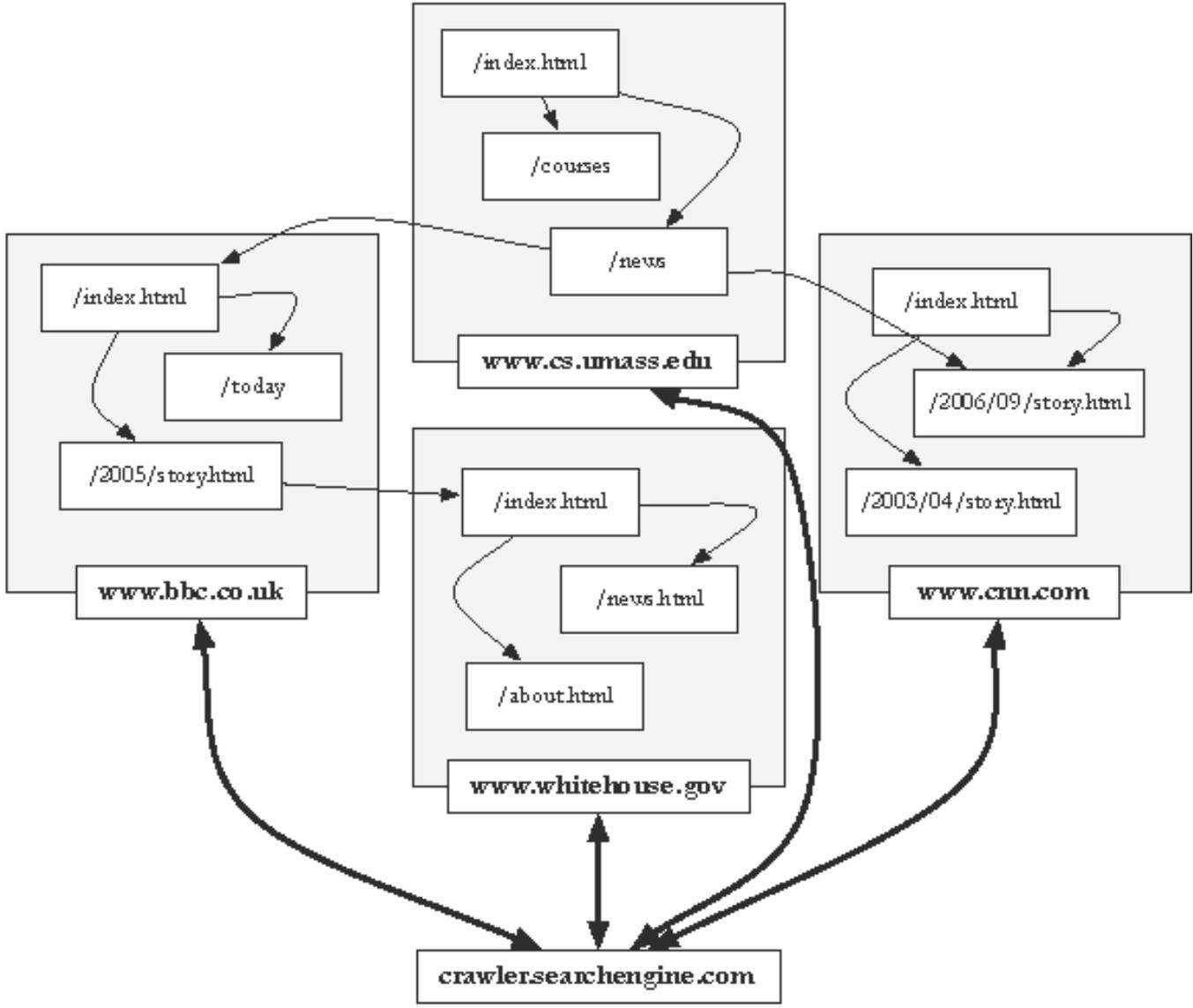


retrieving web pages

- web crawler client program connects to a domain name system (DNS) server
- DNS server translates the hostname into an internet protocol (IP) address
- crawler then attempts to connect to server host using specific port
- after connection, crawler sends an HTTP request to the web server to request a page
 - usually a GET request
 - e.g., GET /csinfo/people.html HTTP/1.0



crawling the web



web crawler

- starts with a set of seeds, which are a set of URLs given to it as parameters
- seeds are added to a URL request queue
- crawler starts fetching pages from the request queue
- downloaded pages are parsed to find **link tags** that might contain other useful URLs to fetch
- new URLs added to the crawler's request queue, or frontier
- continue until no more new URLs or disk full



web crawling

- web crawlers spend a lot of time waiting for responses to requests
- to reduce this inefficiency, web crawlers use **threads** and fetch hundreds of pages at once
- crawlers could potentially flood sites with requests for pages
- to avoid this problem, web crawlers use **politeness policies**
 - e.g., delay between requests to same web server



controlling crawling

- even crawling a site slowly will anger some web server administrators, who object to any copying of their data
- robots.txt file can be used to control crawlers

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

```
User-agent: FavoredCrawler  
Disallow:
```

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



simple crawler thread

```
procedure CRAWLERTHREAD(frontier)
  while not frontier.done() do
    website ← frontier.nextSite()
    url ← website.nextURL()
    if website.permitsCrawl(url) then
      text ← retrieveURL(url)
      storeDocument(url, text)
      for each url in parse(text) do
        frontier.addURL(url)
      end for
    end if
    frontier.releaseSite(website)
  end while
end procedure
```



freshness

- web pages are constantly being added, deleted, and modified
- web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the **freshness** of the document collection
 - **stale** copies no longer reflect the real contents of the web pages



freshness

- HTTP protocol has a special request type called HEAD that makes it easy to check for page changes
 - returns information about page, not page itself
 - Last-Modified: indicates the last time the page content was changed

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

```
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
```

```
Server response: ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

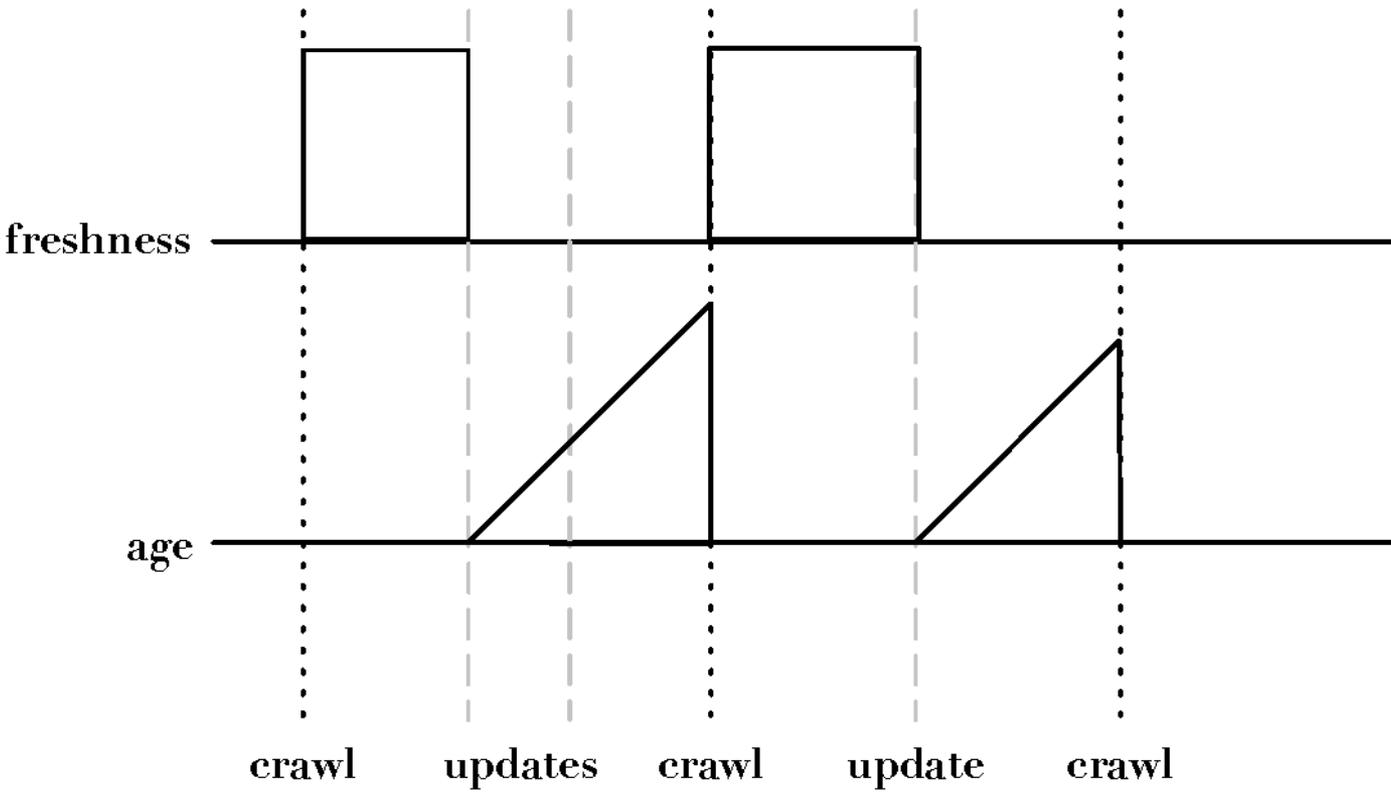


freshness

- not possible to constantly check all pages
 - must check important pages and pages that change frequently
- **freshness** is the proportion of pages that are currently fresh
- optimizing for this metric can lead to bad decisions, such as not crawling popular sites
- **age** is a better metric
 - page has age 0 until it is changed, and then its age grows until the page is crawled again



freshness vs. age



an inventory control problem!



age

- $P(\text{page changed at time } x)$: probability that the page actually changed at time x
- expected age of a page t days after it was last crawled:

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

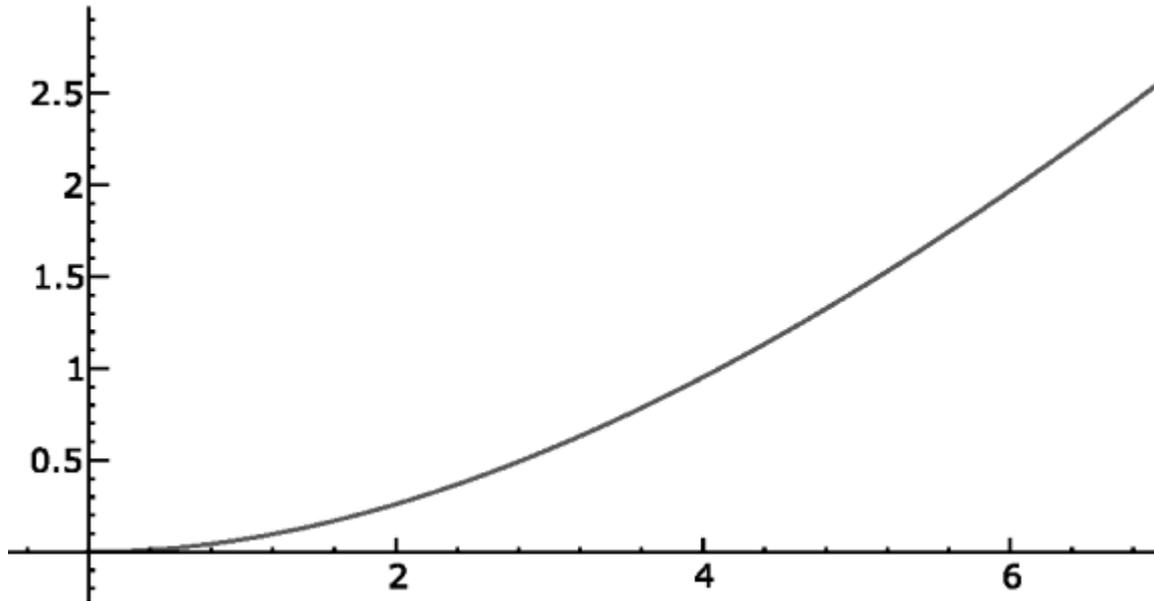
- web page updates follow the Poisson distribution on average
 - time until the next update is governed by an exponential distribution

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



age

- older a page gets, the more it costs not to crawl it
 - e.g., expected age with mean change frequency $\lambda = 1/7$ (one change per week)



focused crawling

- attempts to download only those pages that are about a **particular topic**
 - used by vertical search applications
- rely on the fact that pages about a topic tend to have links to other pages on the same topic
 - popular pages for a topic are typically used as seeds
- crawler uses text classifier to decide whether a page is on topic



deep web

- sites that are difficult for a crawler to find are collectively referred to as the deep (or hidden) web
 - much larger than conventional web: 550B (as of 2001)
- 3 broad categories:
 - private sites
 - no incoming links, or may require log in with a valid account
 - form results
 - sites that can be reached only after entering some data into a form
 - scripted pages
 - pages that use JavaScript, Flash, or another client-side language to generate links



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">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/items?item=truck</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/items?item=bicycle</loc>
    <changefreq>daily</changefreq>
  </url>
</urlset>
```



distributed crawling

- 3 reasons to use multiple computers for crawling
 - helps to put the crawler closer to the sites it crawls
 - reduces the number of sites the crawler has to remember
 - reduces computing resources required
- distributed crawler uses a hash function to assign URLs to crawling computers
 - hash function should be computed on the host part of each URL



desktop crawls

- used for desktop search and enterprise search
- differences to web crawling:
 - much easier to find the data
 - responding quickly to updates is more important
 - must be conservative in terms of disk and CPU usage
 - many different document formats
 - data privacy very important



document feeds

- many documents are **published**
 - created at a fixed time and rarely updated again
 - e.g., news articles, blog posts, press releases, email
- published documents from a single source can be ordered in a sequence called a **document feed**
 - new documents found by examining the end of the feed



document feeds

- 2 types:
 - **push feed** alerts the subscriber to new documents
 - **pull feed** requires the subscriber to check periodically for new documents
- most common format for pull feeds is called **RSS**
 - Really Simple Syndication, RDF Site Summary, Rich Site Summary, or ...
 - **ttl** tag (time to live)
 - amount of time (in minutes) contents should be cached
 - RSS feeds are accessed like web pages
 - using HTTP GET requests to web servers that host them
 - easy for crawlers to parse
 - easy to find new information
 - cf: atom



RSS example

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Search Engine News</title>
    <link>http://www.search-engine-news.org/</link>
    <description>News about search engines.</description>
    <language>en-us</language>
    <pubDate>Tue, 19 Jun 2008 05:17:00 GMT</pubDate>
    <ttl>60</ttl>

    <item>
      <title>Upcoming SIGIR Conference</title>
      <link>http://www.sigir.org/conference</link>
      <description>The annual SIGIR conference is coming!
        Mark your calendars and check for cheap
        flights.</description>
      <pubDate>Tue, 05 Jun 2008 09:50:11 GMT</pubDate>
      <guid>http://search-engine-news.org#500</guid>
    </item>
    ...
    <item>
      <title>New Search Engine Textbook</title>
      <link>http://www.cs.umass.edu/search-book</link>
      <description>A new textbook about search engines
        will be published soon.</description>
      <pubDate>Tue, 05 Jun 2008 09:33:01 GMT</pubDate>
      <guid>http://search-engine-news.org#499</guid>
    </item>
  </channel>
</rss>
```



conversion problem

- text is stored in hundreds of incompatible file formats
 - e.g., raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF
- other types of files also important
 - e.g., PowerPoint, Excel
- typically use a conversion tool
 - converts the document content into a tagged text format such as HTML or XML
 - retains some of the important formatting information



character encoding

- a mapping between bits and glyphs
 - i.e., getting from bits in a file to characters on a screen
 - can be a major source of incompatibility
- ASCII is basic character encoding scheme for english
 - encodes 128 letters, numbers, special characters, and control characters in 7 bits, extended with an extra bit for storage in bytes



character encoding

- other languages can have many more glyphs
 - e.g., chinese has more than 40,000 characters, with over 3,000 in common use
- many languages have multiple encoding schemes
 - e.g., CJK (Chinese-Japanese-Korean) family of east asian languages, hindi, arabic
 - must specify encoding
 - can't have multiple languages in one file
- Unicode developed to address encoding problems



Unicode

- single mapping **from numbers to glyphs** that attempts to include all glyphs in common use in all known languages
 - many ways to translate Unicode numbers to glyphs: UTF-8, UTF-16, UTF-32
- proliferation of encodings comes from a need for compatibility and to save space
 - UTF-8 uses one byte for English (ASCII), as many as 4 bytes for some traditional Chinese characters
 - variable length encoding, more difficult to do string operations
 - UTF-32 uses 4 bytes for every character
- many applications use **UTF-32 for internal text encoding** (fast random lookup) and **UTF-8 for disk storage** (less space)



UTF-8

Decimal	Hexadecimal	Encoding			
0–127	0–7F	0xxxxxxx			
128–2047	80–7FF	110xxxxx	10xxxxxx		
2048–55295	800–D7FF	1110xxxx	10xxxxxx	10xxxxxx	
55296–57343	D800–DFFF	Undefined			
57344–65535	E000–FFFF	1110xxxx	10xxxxxx	10xxxxxx	
65536–1114111	10000–10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- e.g., Greek letter pi (π) is Unicode symbol number 960
- to encode decimal 2047, we only need 11 bits
- in binary, `00000011 11000000` (3C0 in hexadecimal)
- final encoding: `11001111 10000000` (CF80 in hexadecimal)



storing the documents

- many reasons to store converted document text
 - no need for crawling again when page is not updated
 - provides efficient access to text for snippet generation, information extraction, etc.
- database systems can provide document storage for some applications
 - web search engines use customized document storage systems



storing the documents

- requirements for document storage system:
 - random access
 - request the content of a document based on its URL
 - hash function based on URL is typical
 - compression and large files
 - reducing storage requirements and efficient access
 - update
 - handling large volumes of new and modified documents
 - adding new anchor text



large files

- store many documents in large files, rather than each document in a file
 - avoids overhead in opening and closing files
 - reduces seek time relative to read time
- compound documents formats
 - used to store multiple documents in a file
 - e.g., TREC Web



TREC Web Format

```
<DOC>
<DOCNO>WTX001-B01-10</DOCNO>
<DOCHDR>
http://www.example.com/test.html 204.244.59.33 19970101013145 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:13 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Tropical Fish Store</TITLE>
Coming soon!
</HTML>
</DOC>
<DOC>
<DOCNO>WTX001-B01-109</DOCNO>
<DOCHDR>
http://www.example.com/fish.html 204.244.59.33 19970101013149 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:19 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Fish Information</TITLE>
This page will soon contain interesting
information about tropical fish.
</HTML>
</DOC>
```



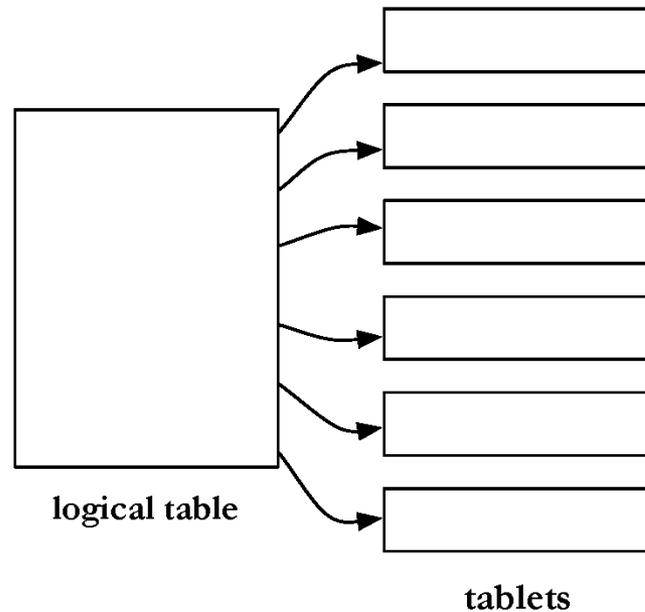
compression

- text is highly redundant (or predictable)
- compression techniques exploit this redundancy to make files smaller **without losing any of the content**
- popular algorithms can compress HTML and XML text by 80%
 - e.g., DEFLATE (zip, gzip) and LZW (UNIX compress, PDF)
 - may compress large files in blocks to make access faster



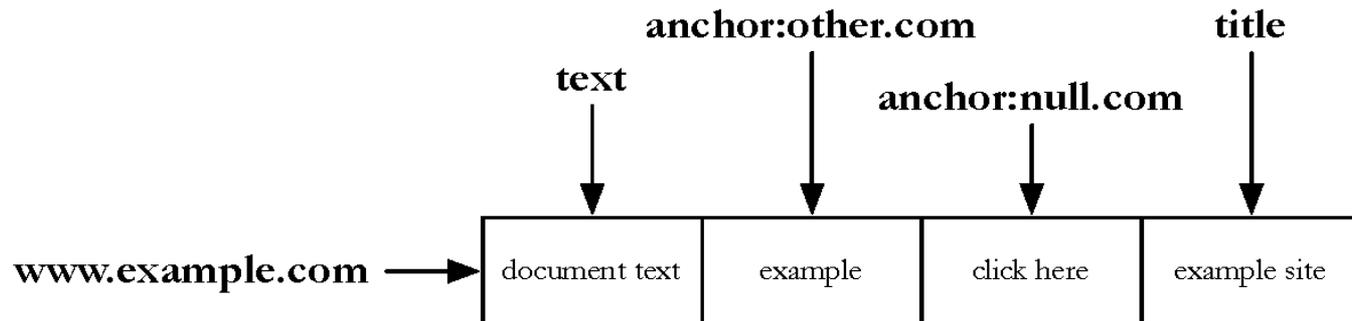
BigTable

- Google's document storage system
 - customized for storing, finding, and updating web pages
 - handles large collection sizes using inexpensive computers



BigTable

- tablets are stored in a replicated file system that is accessible by all BigTable servers
- logically organized into rows
- a row stores data for a single web page
- BigTable can have a huge number of columns per row
- rows are partitioned into tablets based on their row keys
 - simplifies determining which server is appropriate



detecting duplicates

- duplicate and near-duplicate documents occur in many situations
 - copies, versions, plagiarism, spam, mirror sites
 - 30% of the web pages in a large crawl are exact or near duplicates of pages in the other 70%
- duplicates consume significant resources during crawling, indexing, and search
 - little value to most users



duplicate detection

- exact duplicate detection is relatively easy
- checksum techniques
 - checksum: a value that is computed based on the content of the document
 - e.g., sum of the bytes in the document file

T	r	o	p	i	c	a	l		f	i	s	h	<i>Sum</i>
54	72	6F	70	69	63	61	6C	20	66	69	73	68	508

- possible for files with different text to have same checksum
- functions such as a **cyclic redundancy check (CRC)**, have been developed that consider the positions of the bytes



near-duplicate detection

- more challenging task
 - are web pages with same text context but different advertising or format near-duplicates?
- a near-duplicate document is defined using a **threshold** value for some similarity measure between pairs of documents
 - e.g., document D1 is a near-duplicate of document D2 if more than 90% of the words in the documents are the same



near-duplicate detection

- search scenario
 - find near-duplicates of a document D
 - $O(N)$ comparisons required
- discovery:
 - find **all pairs of near-duplicate documents** in the collection
 - $O(N^2)$ comparisons
- IR techniques are effective for search scenario
- for discovery, other techniques used to generate compact representations



fingerprints

1. The document is parsed into words. Non-word content, such as punctuation, HTML tags, and additional whitespace, is removed.
2. The words are grouped into contiguous *n-grams* for some *n*. These are usually overlapping sequences of words, although some techniques use non-overlapping sequences.
3. Some of the n-grams are selected to represent the document.
4. The selected n-grams are hashed to improve retrieval efficiency and further reduce the size of the representation.
5. The hash values are stored, typically in an inverted index.
6. Documents are compared using overlap of fingerprints



fingerprint example

Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

(a) Original text

tropical fish include, fish include fish, include fish found, fish found in, found in tropical, in tropical environments, tropical environments around, environments around the, around the world, the world including, world including both, including both freshwater, both freshwater and, freshwater and salt, and salt water, salt water species

(b) 3-grams

938 664 463 822 492 798 78 969 143 236 913 908 694 553 870 779

(c) Hash values

664 492 236 908

(d) Selected hash values using $0 \bmod 4$

select all n -grams whose hash value modulo p is zero



simhash

- similarity comparisons using word-based representations more effective at finding near-duplicates
 - problem is efficiency
- simhash combines the advantages of the word-based similarity measures with the efficiency of fingerprints based on hashing
- similarity of two pages as measured by the cosine correlation measure is proportional to the number of bits that are the same in the simhash fingerprints



simhash

1. Process the document into a set of features with associated weights. We will assume the simple case where the features are words weighted by their frequency.
2. Generate a hash value with b bits (the desired size of the fingerprint) for each word. The hash value should be unique for each word.
3. In b -dimensional vector V , update the components of the vector by adding the weight for a word to every component for which the corresponding bit in the word's hash value is 1, and subtracting the weight if the value is 0.
4. After all words have been processed, generate a b -bit fingerprint by setting the i th bit to 1 if the i th component of V is positive, or 0 otherwise.

a web page is defined as a near-duplicate of another page if simhash fingerprints agree on more than prespecified x bits



simhash example

Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

(a) Original text

tropical 2 fish 2 include 1 found 1 environments 1 around 1 world 1
including 1 both 1 freshwater 1 salt 1 water 1 species 1

(b) Words with weights

tropical	01100001	fish	10101011	include	11100110
found	00011110	environments	00101101	around	10001011
world	00101010	including	11000000	both	10101110
freshwater	00111111	salt	10110101	water	00100101
species	11101110				

(c) 8 bit hash values

1 -5 9 -9 3 1 3 3

(d) Vector V formed by summing weights

1 0 1 0 1 1 1 1

(e) 8-bit fingerprint formed from V



removing noise

- many web pages contain text, links, and pictures that are not directly related to the main content of the page
- this additional material is mostly **noise** that could negatively affect the ranking of the page
- techniques have been developed to **detect the content blocks** in a web page
 - non-content material is either ignored or reduced in importance in the indexing process



noise example

The image shows a screenshot of the CNN.com website from 2007. The main article is titled "Aquarium plays whale shark matchmaker" and is dated Monday, June 5, 2006. The article text is as follows:

Two females flown 8,000 miles for double date in Atlanta

Monday, June 5, 2006, Posted: 9:28 p.m. EDT (11:28 GMT)

ATLANTA, Georgia (CNN) — Ralph and Norton, meet Alice and Trixie.

The Georgia Aquarium's two male whale sharks got some female companionship on Saturday, when they were joined by two females transported to Atlanta from Taipei, Taiwan.

Researchers are hoping the sharks will mate.

The females — 11 feet and 14 feet long — were flown more than 8,000 miles by UPS, which reconfigured a company B-747 freighter with advanced marine life support systems to carry them. ([Watch what it took to get the sharks together — 1:55](#))

The pilot said they treated the massive fish like first-class passengers.

"As we were doing the descent, we asked to start down a little sooner to make a nice shallower descent, to not make things too uncomfortable back there for the whale sharks," UPS pilot Capt. Bob Crum said.

The plane's center of balance was carefully planned, according to a statement from the aquarium, and veterinarians accompanied the sharks.

The delivery company also brought the two males to Atlanta, where researchers can study the whale sharks' behavior, breeding and development.

The whale sharks — named after the main characters in the 1950s sitcom "The Honeymooners" — were delivered to the aquarium in special transportation containers.

The Georgia Aquarium, which opened in November, is the world's largest aquarium. It was a \$250 million gift to Georgia from Bernie Marcus, co-founder of The Home Depot and his wife, Bill, through the Marcus Foundation.

It is the only aquarium outside of Asia to showcase whale sharks, which are the fish on Earth.

The aquarium's 6.2-million gallon "Ocean Voyager" tank can hold up to six whale sharks, and there's room for the whale sharks to start a family.

Below the article, there are sections for "Story Tools", "SPACE" (with sub-sections like "Astronauts prepare for third spacewalk", "Astronomers vie to make biggest telescope", "NASA to beam Beatles song to North Star", "U.S. plans for falling satellite"), and "TOP STORIES" (with sub-sections like "Russians choose Putin's successor", "Iran's president makes landmark visit to Iraq", "Israel PM: Attacks on militants go on", "Cable arrested in abandoned baby case").

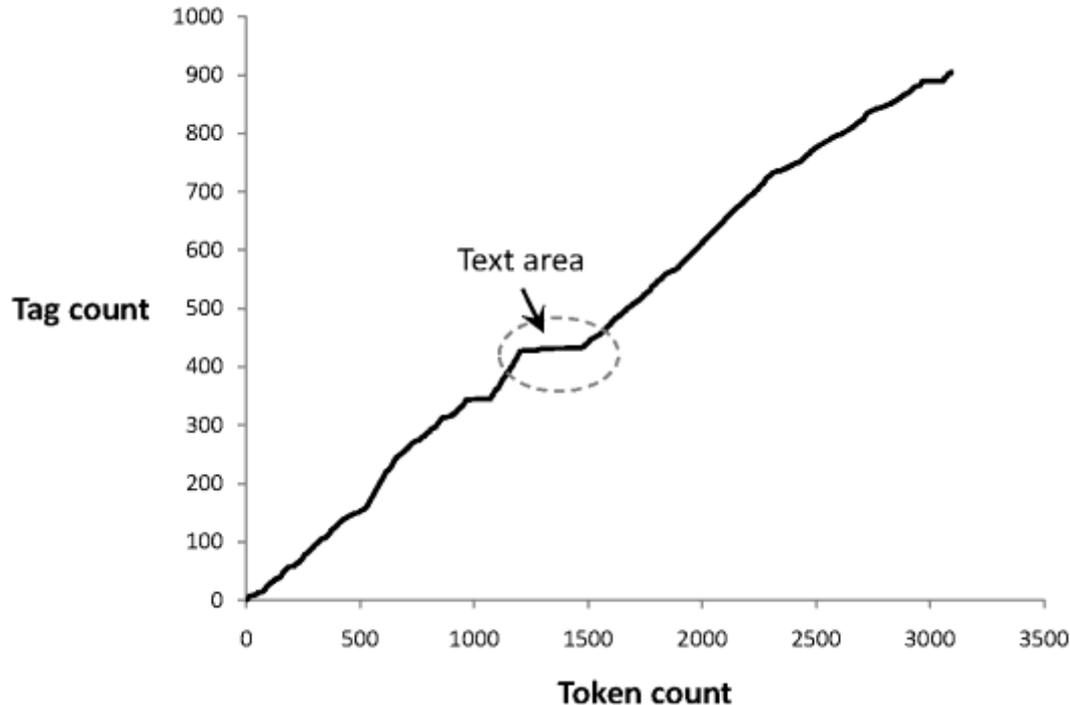
A black box highlights the main article text, and an arrow points to it with the label "Content block".

Content block



finding content Blocks

- **cumulative** distribution of tags in the example web page



- main text content of the page corresponds to the “plateau” in the middle of the distribution

finding content blocks

- represent a web page as a **sequence of bits**, where $b_n = 1$ indicates that the n th token is a tag
- optimization problem where we **find values of i and j to maximize both the number of tags below i and above j and the number of non-tag tokens between i and j**
- i.e., maximize

$$\sum_{n=0}^{i-1} b_n + \sum_{n=i}^j (1 - b_n) + \sum_{n=j+1}^{N-1} b_n$$



finding content blocks

- other approaches use DOM structure and visual (layout) features

