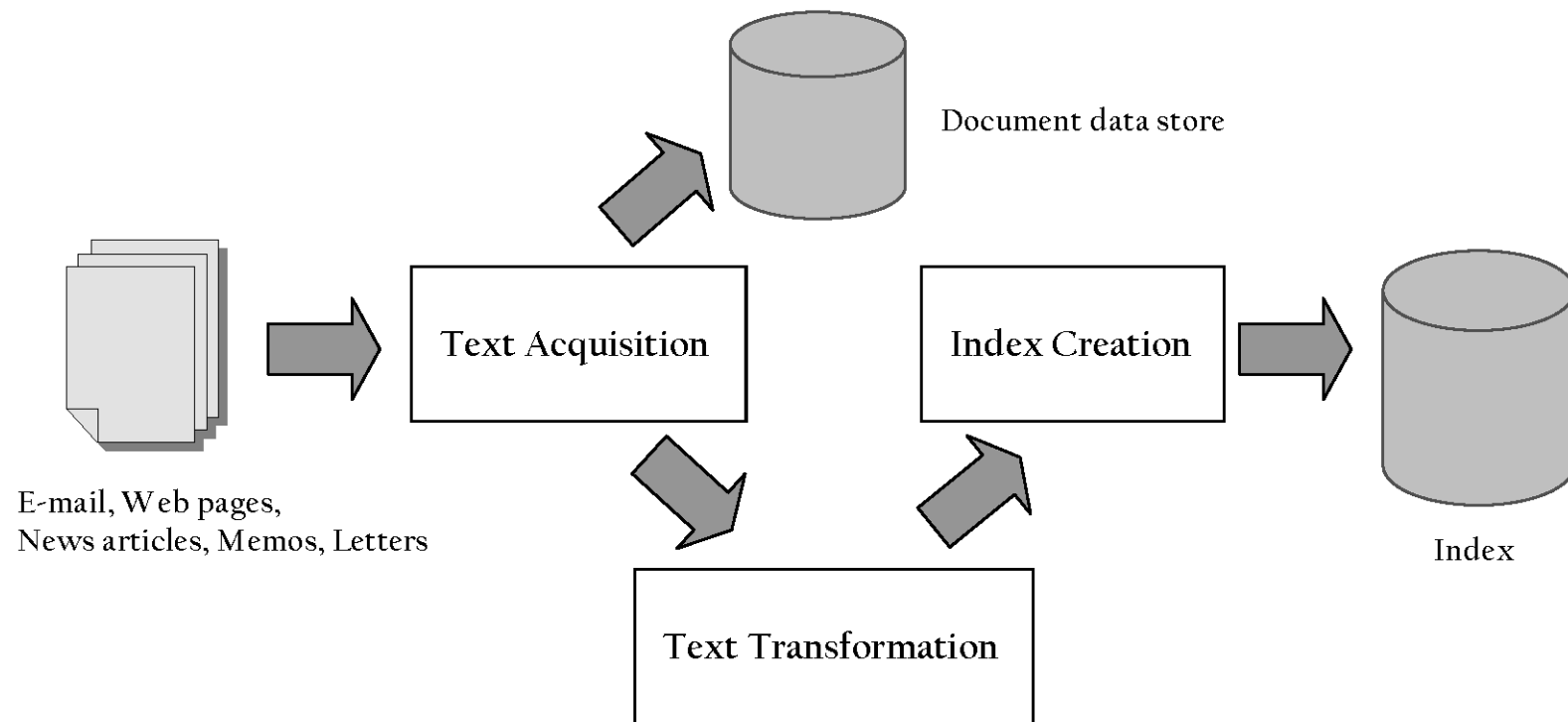


Search Engines

Information Retrieval in Practice

Indexing Process



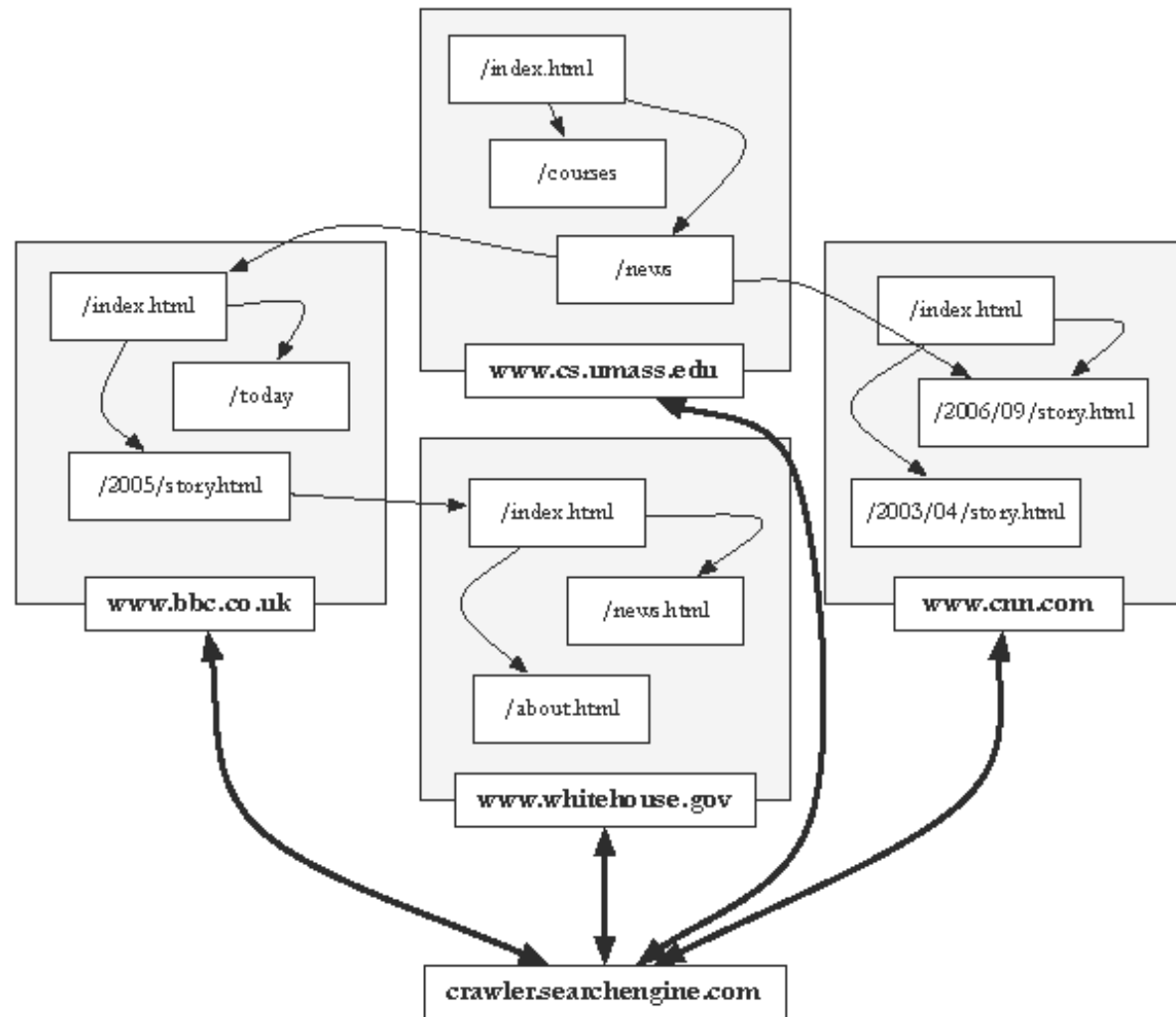
Web Crawler

- Finds and downloads web pages automatically
 - provides the collection for searching
- Web is huge and constantly growing (scalable, efficient)
- Web pages are constantly changing (continuous)
- Web is not under the control of search engine providers (polite)
- Crawlers also used for other types of data (extensible)

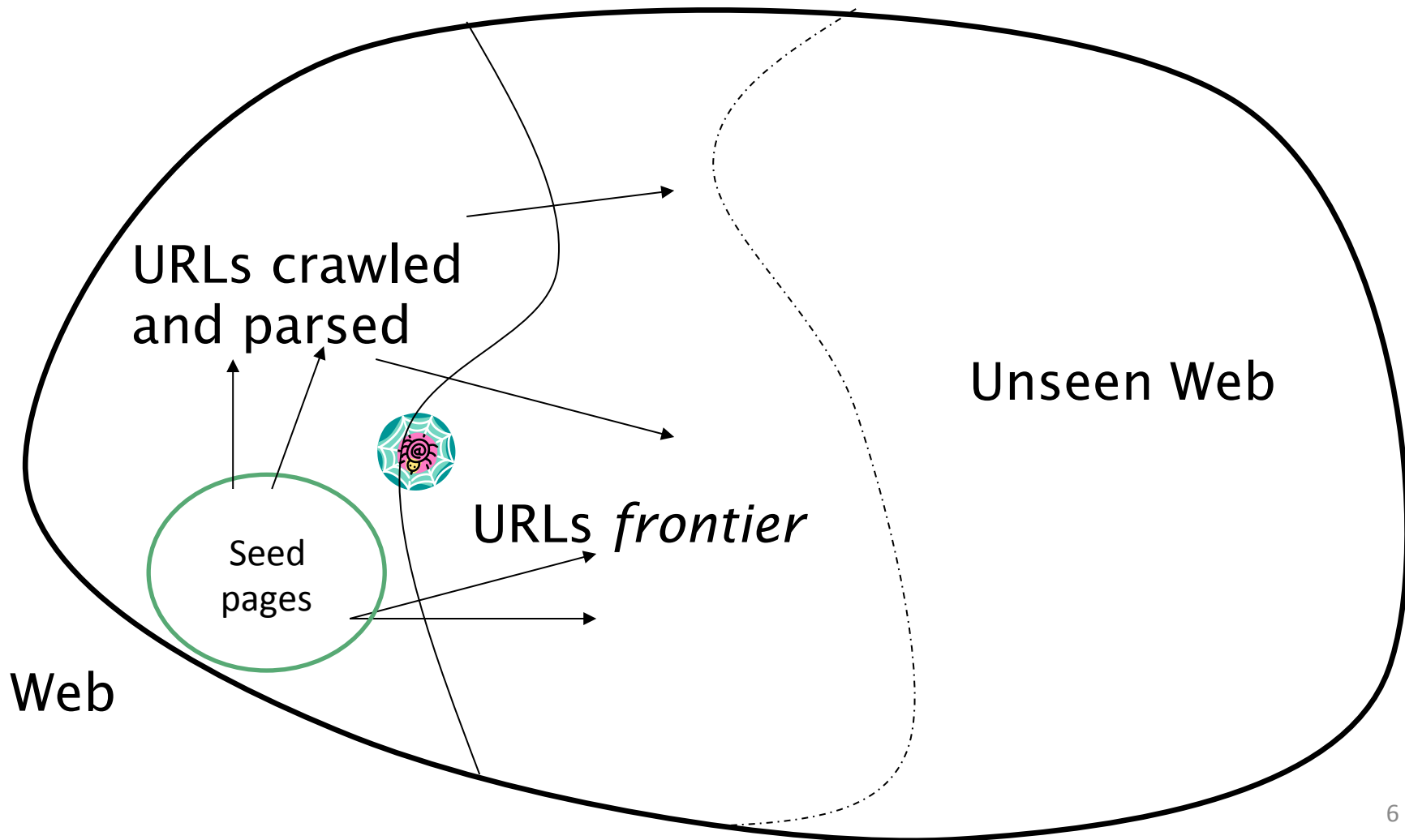
Web Crawler

- 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

Crawling the Web



Crawling picture



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.,

http://www.cs.umass.edu/csinfo/people.html

The diagram illustrates the components of the URL `http://www.cs.umass.edu/csinfo/people.html`. Below the URL, three labels are positioned: **scheme** under `http`, **hostname** under `www.cs.umass.edu`, and **resource** under `/csinfo/people.html`. Double-headed arrows connect each label to its corresponding part of the URL.

scheme **hostname** **resource**

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

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*
 - *Explicit or Implicit*
 - e.g., one page at a time and 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/
```

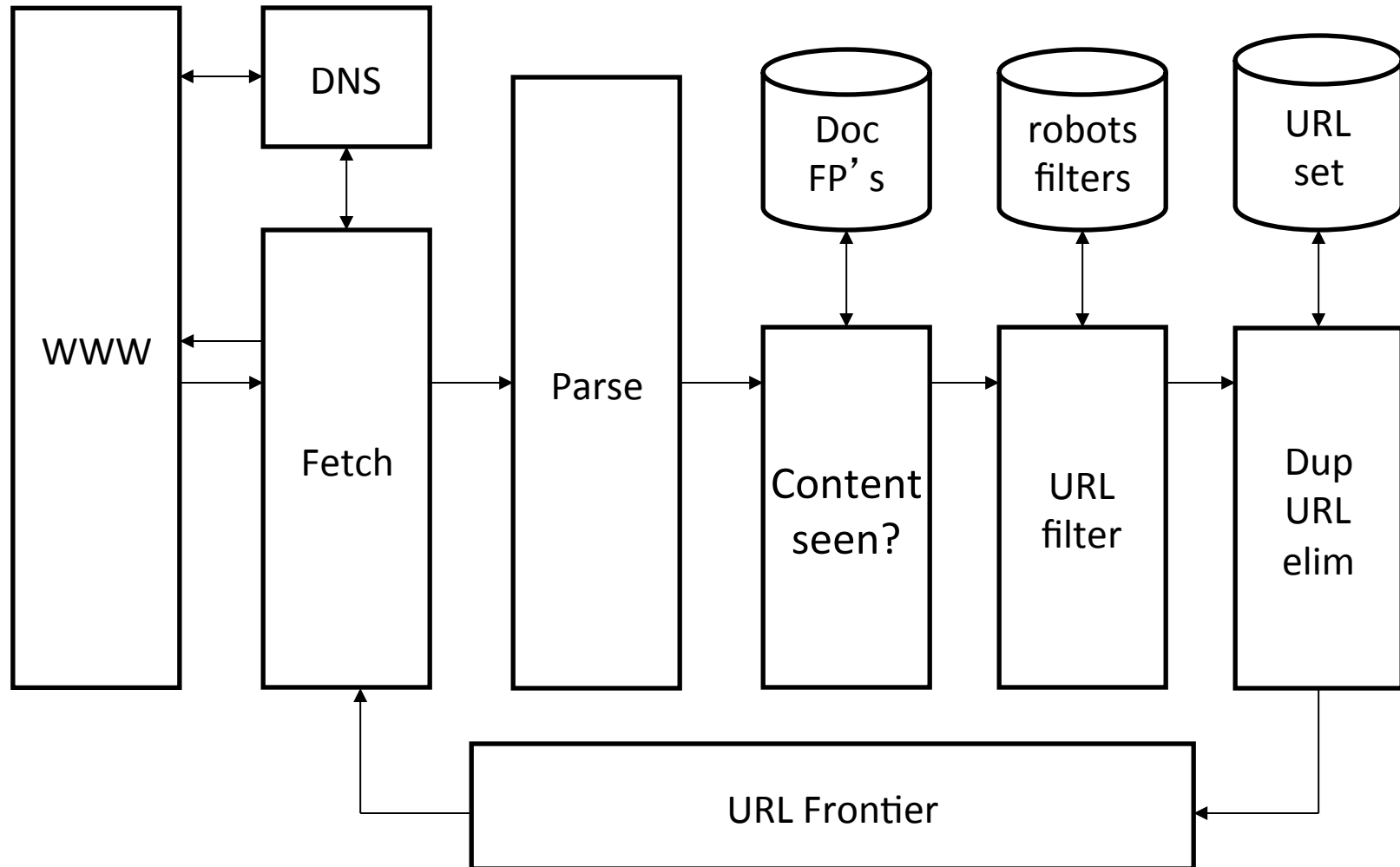
```
User-agent: FavoredCrawler  
Disallow:
```

```
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
```

Basic crawl architecture



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

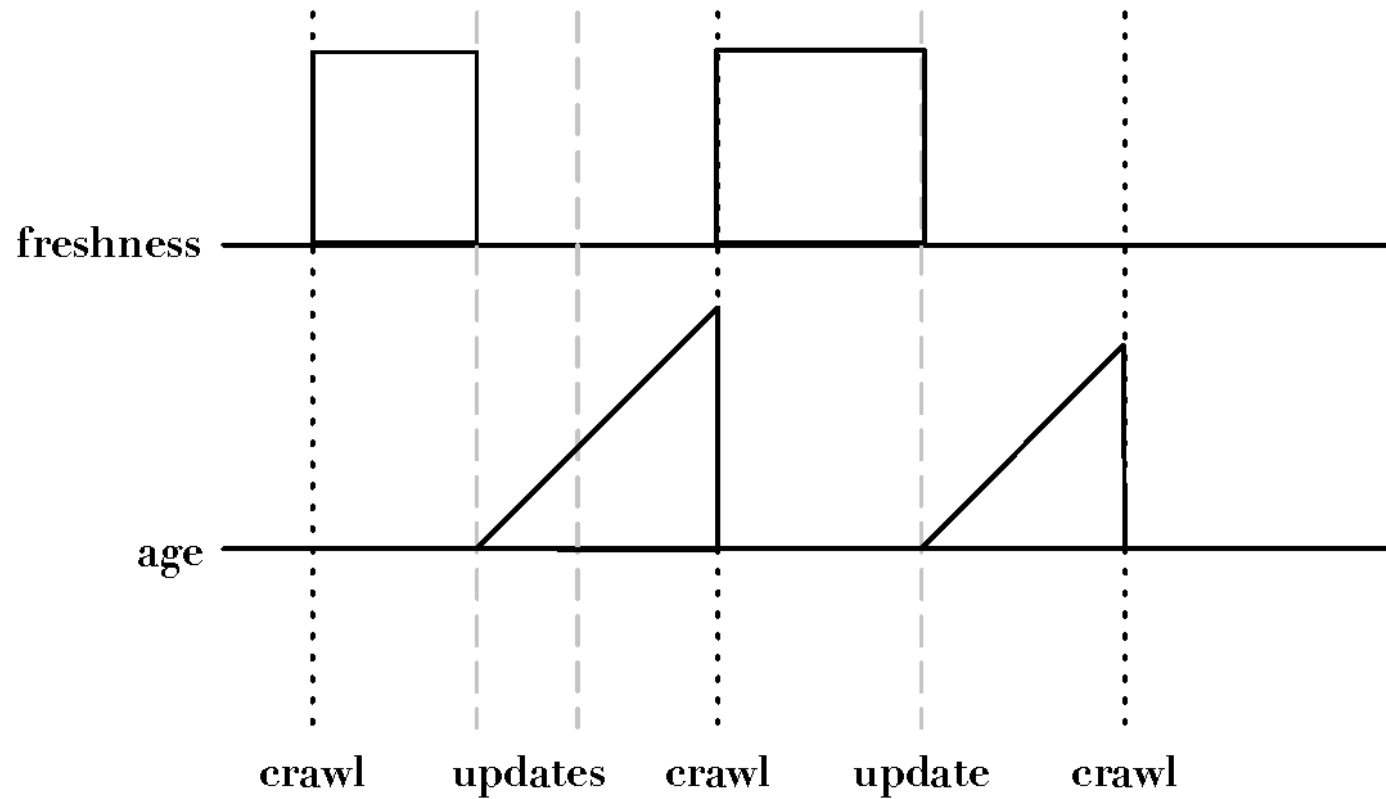
Client request: HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu

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 fresh
- Optimizing for this metric can lead to bad decisions, such as not crawling popular sites
- *Age* is a better metric

Freshness vs. Age



Age

- Expected age of a page t days after it was last crawled:

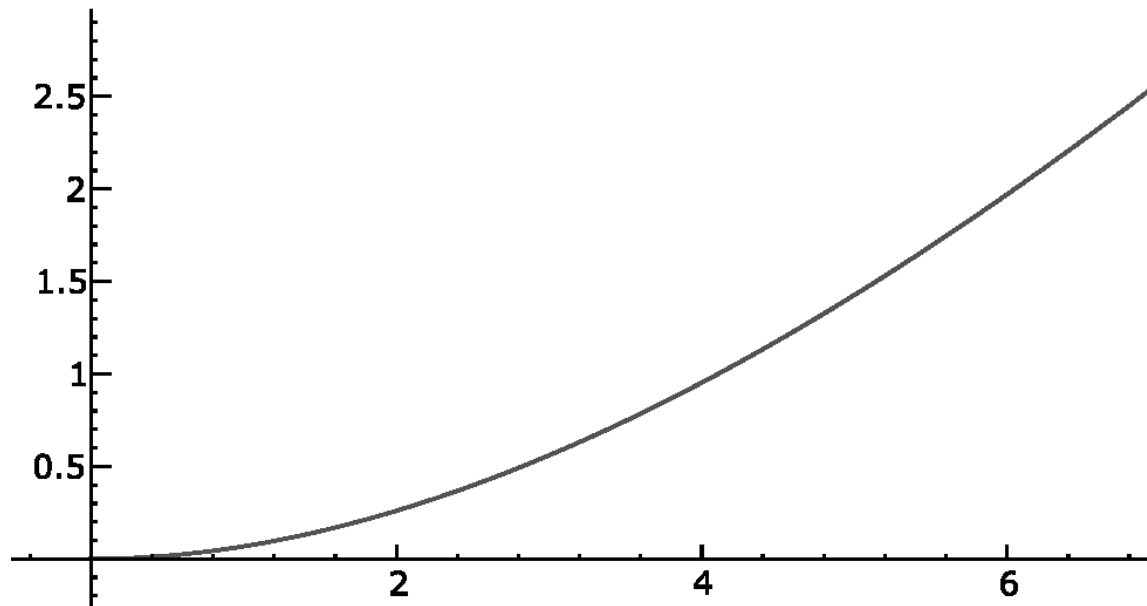
$$\text{Age}(\lambda, t) = \int_0^t P(\text{page changed at time } x)(t - x)dx$$

- 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
- Three 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

Surfacing the Deep Web

- Pre-compute all interesting form submissions each HTML form
- Each form submission corresponds to a distinct URL
- Add URLs for each form submission into search engine index

Form Processing 101

FIND U.S. LOCATIONS
Please enter information in one or more of the fields below. City searches require a state.


STORE

CITY

STATE

ZIP

WITHIN



```
<form action=http://www.borders.com/locator method=GET>  
<select name=store><option .../>... </select>  
...  
<input name=zip type=text/>  
<input name=search type=submit value=Go/>  
<input name=site type=hidden value=homepage/>  
</form>
```

on submit

URL: *http://www.borders.com/locator?store=All&city=&state=*
&zip=94043&within=25&search=Go&site=homepage

GET and POST: types of HTML forms

Only GETs can be surfaced

Surfacing Challenges

1. Predicting the appropriate values for text inputs
Valid input values are required for retrieving data
Ingredients in recipes.com and zipcodes in borderstores.com
2. Predicting the correct input combinations
Generating all possible URLs is wasteful + unnecessary
Cars.com has ~500K listings, but 250M possible queries

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>
```

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

- Two types:
 - A *push feed* alerts the subscriber to new documents
 - A *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 ...

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>
```

RSS Example

...

```
<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>
```

RSS

- `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

Distributed Crawling

- Three 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

Conversion

- 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 character encoding is 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
- Unicode is a mapping between numbers and glyphs
 - e.g., UTF-8, UTF-16, UTF-32

Unicode

- 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)

Unicode

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
- In binary, 00000011 11000000 (3C0 in hexadecimal)
- Final encoding is **11001111 10000000** (CF80 in hexadecimal)

Storing the Documents

- Many reasons to store converted document text
 - saves crawling time 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
 - Many documents per file
 - Update
 - handling large volumes of new and modified documents
 - adding new anchor text

Compression

- Text is highly redundant (or predictable)
- Compression techniques exploit this redundancy to make files smaller without losing any of the content
- Compression of indexes covered later
- 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

Large Files

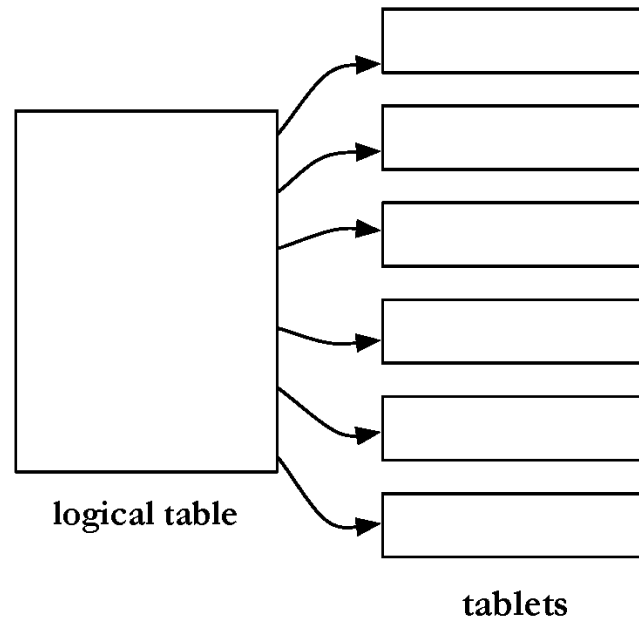
- Store many documents in large files, rather than each document in a file
 - avoids overhead in opening and closing files
 - reduces seek time relative to read time
- Compound documents formats
 - used to store multiple documents in a file
 - e.g., TREC Web

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>
```

BigTable

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

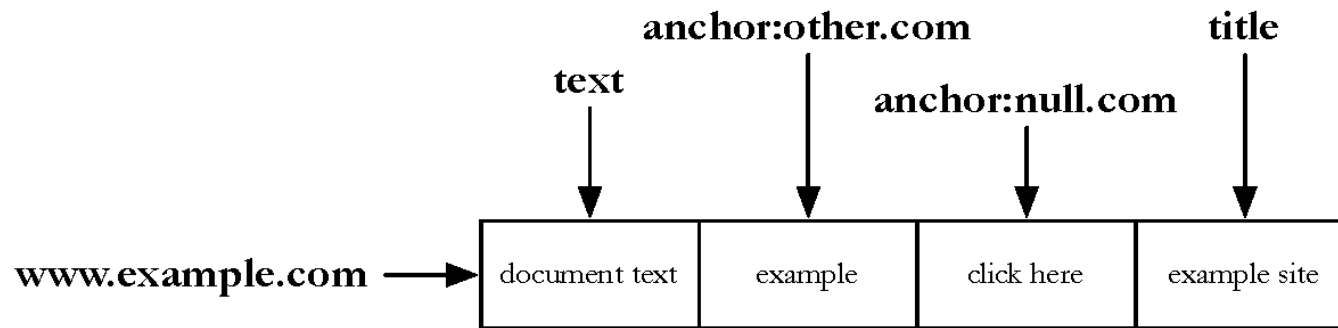


BigTable

- No query language, no complex queries to optimize
- Only row-level transactions
- Tablets are stored in a replicated file system that is accessible by all BigTable servers
- Any changes to a BigTable tablet are recorded to a transaction log, which is also stored in a shared file system
- If any tablet server crashes, another server can immediately read the tablet data and transaction log from the file system and take over

BigTable

- Logically organized into rows
- A row stores data for a single web page



- Combination of a row key, a column key, and a timestamp point to a single *cell* in the row

BigTable

- BigTable can have a huge number of columns per row
 - all rows have the same column groups
 - not all rows have the same columns
 - important for reducing disk reads to access document data
- 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
 - Remove or treat them as a group

Duplicate Detection

- *Exact* duplicate detection is relatively easy
- *Checksum* techniques
 - A checksum is 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:*
 - 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 $0 \bmod 4$

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.

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

CNN.com Member Center Sign In Register International Edition

SEARCH 11:11 AM EST 05/05/2007 Search

Home Page World U.S. Weather Business Sports Analysis Politics Law Technology Science & Space Health Entertainment Offbeat Travel Education Special Reports Video Audio I-Report

IMPACT WORLD TAKE ACTION

SERVICES E-mail RSS Podcasts Mobile CNN Pipeline SEARCH WEB CHN.COM Search

SCIENCE & SPACE

Aquarium plays whale shark matchmaker

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

Monday, June 5, 2006, Posted: 5:28 p.m. EDT (21:30 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 9-747 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 shallow 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 largest 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.

Image: Alice the whale shark swims into the Ocean Voyager tank at the Georgia Aquarium for the first time.

Image: [Image placeholder]

YOUR E-MAIL ALERTS

Atlanta (Georgia)

Taiwan

ACTIVATE or Create Your Own

Manage Alerts | What is This?

Story Tools

Subscribe to Time for \$1.00

SPACE

Section Page | Video

Astronauts prepare for third spacewalk

Russians choose Putin's successor

Astronomers vie to make biggest

NASA to beam Beatles song to North Star

U.S. plans for falling satellite

Iran's president makes landmark visit to Iraq

Israel PM: Attacks on militants go on

Cable arrested in abandoned baby case

TOP STORIES

Home Page | Video | Most Popular

International Edition Languages CNN TV CNN International Headline News Transcripts Advertise with Us About Us

SEARCH THE WEB WEB TV Search

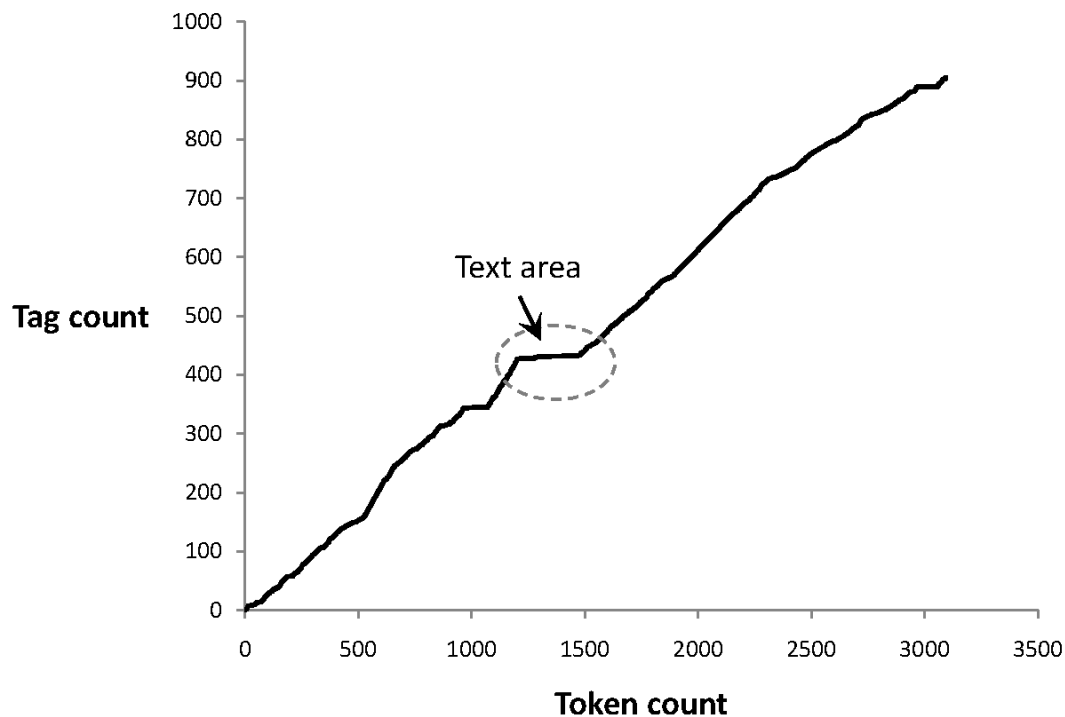
© 2007 Cable News Network. A Time Warner Company. All Rights Reserved. Terms under which this service is provided to you. Read our privacy guidelines. Contact Us. Site Map.

External sites open in new window; not endorsed by CNN.com. Pay service with live and archived video. Learn more.

Download audio news Add RSS headlines

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

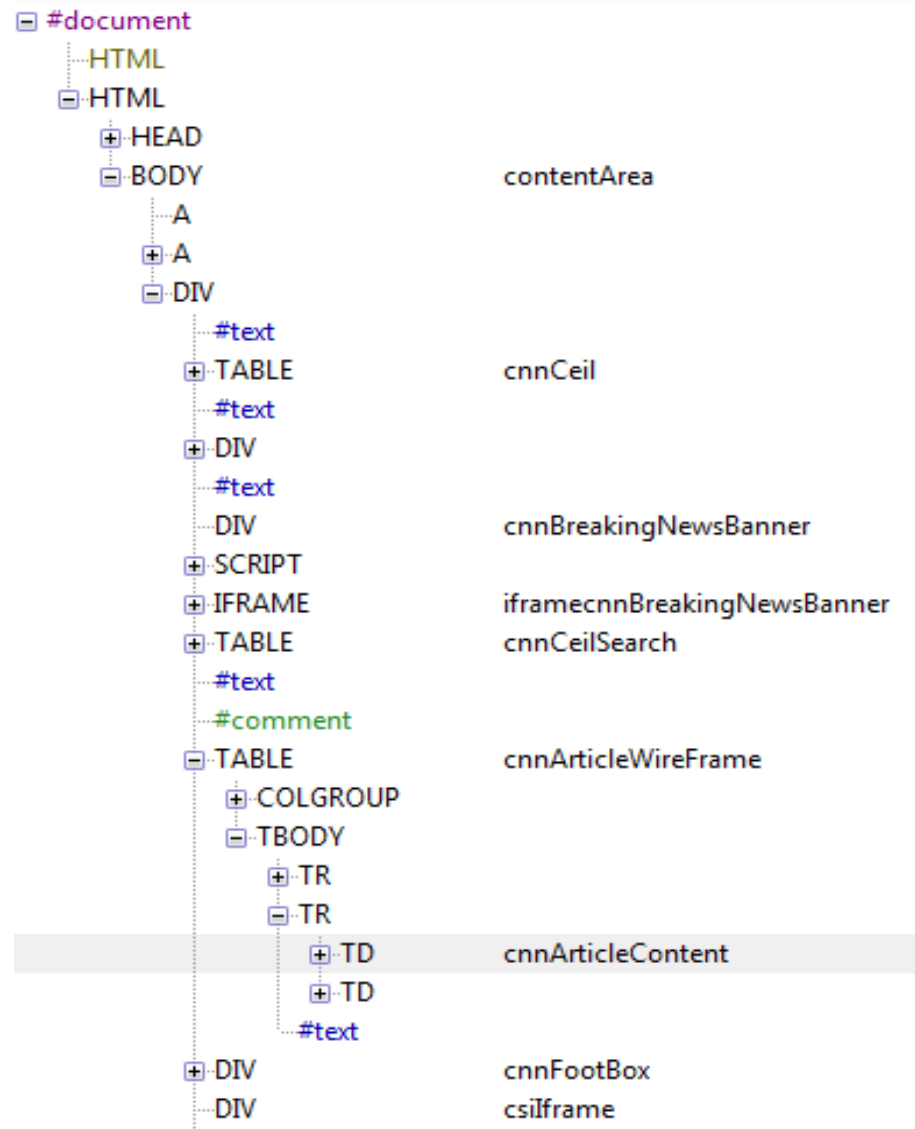
Finding Content Blocks

- 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



ClueWeb09 collection

- About 1 billion web pages
- 10 Languages
- 5 TB compressed (25 TB uncompressed)
- 28 million seed URLs
- <http://boston.lti.cs.cmu.edu/crawler/crawlerstats.html>
- <http://boston.lti.cs.cmu.edu/Data/web08-bst/web08-bst-Sep26-08.pdf>

Apache Nutch

- Open source Web crawler written in Java
- Distributed file system (via Hadoop)
- Supports different document formats
- Scalable, can handle millions of webpages
- Easy to integrate with other Apache projects
- <http://wiki.apache.org/nutch/NutchTutorial>