Chapter IR:IX

IX. Acquisition

Storing Documents

Creating the document store

- ❑ 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 Documents

Requirements for document storage systems

- 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

Storing Documents
Large files

- Store many documents in combined large files, rather than each document in individual small files
  - Avoids overhead in opening and closing files
  - Reduces seek time relative to read time
    - HDD transfer very fast compared to seeking a file
- Compound documents formats
  - Used to store multiple documents in a file
  - TREC Web, WARC

Storing Documents

Example file in TREC Web compound document 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>
```

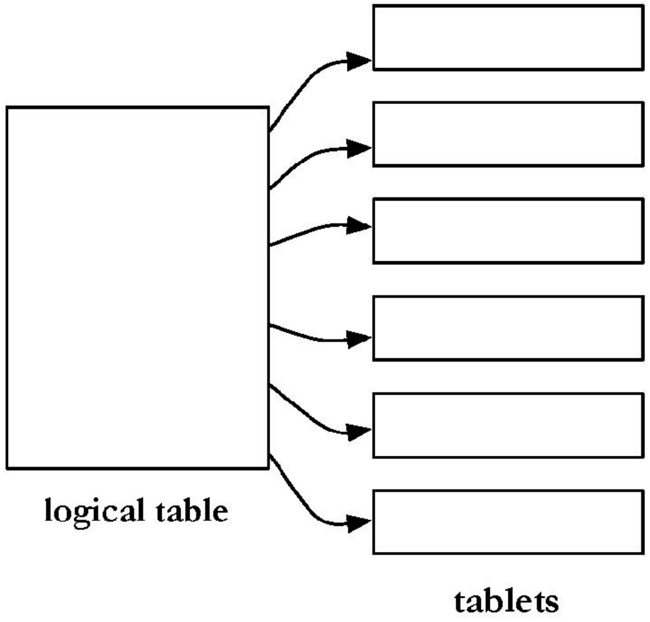
Storing Documents
Compression

- 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 text by 80%
  - DEFLATE (zip, gzip) and LZW (UNIX compress, PDF)
  - May compress large files in blocks to make access faster

Storing Documents

BigTable: Google’s document storage system

- Distributed database system
- Customized for storing, finding, and updating web pages
- Handles large collection sizes using inexpensive computers
- A BigTable instance can be over a petabyte
- Split into small pieces: the tablets



Storing Documents

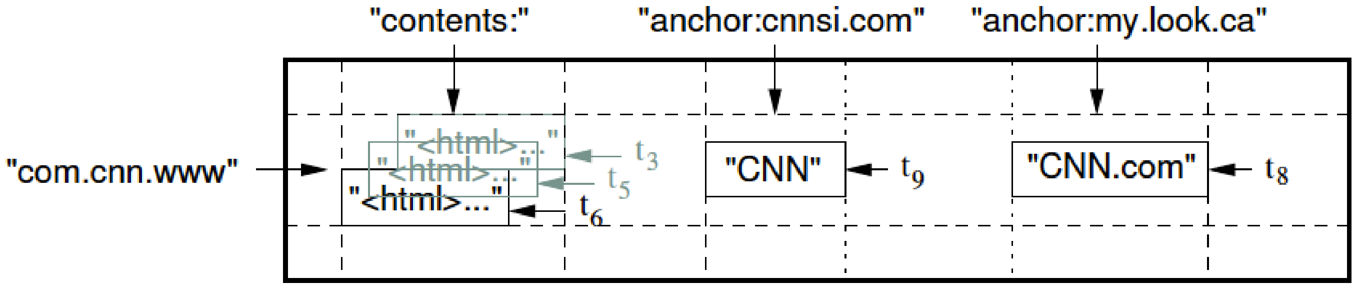
BigTable

- 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

Storing Documents

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 points to a single cell in the row



- 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

Storing Documents

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
  - But add little value to most users

Storing Documents

Detecting Duplicates

- Exact duplicate detection is relatively easy
- Checksum techniques
  - A checksum is a value that is computed based on the content of the documents
  - For instance sum of the bytes in the document file

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

  - Files with different text may have same checksum!
- Techniques such as a cyclic redundancy check (CRC) consider the positions of the bytes

Storing Documents

Near-Duplicate Detection

- More challenging task
  - Are web pages with same text content 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
  - For instance, document d_1 is a near-duplicate of document d_2 if more than 90% of the words in the documents are the same
- Search scenario:
  - Given a document d, find its near-duplicates in a given collection
- Discovery scenario:
  - Given a collection, find all pairs of near-duplicates
- How many comparisons needed in search and in discovery scenario?

Storing Documents

Near-Duplicate Detection

- More challenging task
  - Are web pages with same text content 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
  - For instance, document d_1 is a near-duplicate of document d_2 if more than 90% of the words in the documents are the same
- Search scenario:
  - Given a document d, find its near-duplicates in a given collection
  - $O(n)$ comparisons required (with each of the n corpus documents)
- Discovery scenario:
  - Given a collection, find all pairs of near-duplicates
  - $O(n^2)$ comparisons required

Storing Documents

Near-Duplicate Detection

- IR techniques are effective for the search scenario
- For discovery, other techniques used to generate compact representations
- Fingerprinting:
  1. Parse document into words (remove non-word content like punctuation, HTML tags, etc.)
  2. Group words into contiguous n-grams for some n (usually overlapping sequences of words but also sometimes non-overlapping)
  3. Select some of the n-grams to represent the document
  4. Hash selected n-grams to reduce size and improve retrieval efficiency
  5. Store hash values in an inverted index (details on that later)
  6. Compare documents using overlap of fingerprints

Storing Documents

Fingerprinting example

Original text

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

3-grams

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

Hash values

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

Selected hash values using $0 \bmod 4$

664 492 236 908

Representing 3-grams

`fish include fish, found in tropical, the world including, including both freshwater`

Storing Documents
Fingerprinting

- Finding near-duplicates on the web usually uses 5–10-grams and 64 bit hashes generated using Rabin fingerprinting

  [Broder et al., 1997, *Syntactic clustering of the Web*, Computer Networks 29(8–13): 1157–1166]

- As fingerprints capture not all information, there occur detecting errors
- Often fingerprinting for candidate retrieval and then deeper analysis with for instance cosine similarity
- Cosine similarity is a word-based representation (more on that topic later); problem is efficiency and that's why candidate retrieval first
- Recent `simhash` technique combines the effectiveness of word-based similarity with efficiency of fingerprinting

  [Moses Charikar, 2002, *Similarity estimation techniques from rounding algorithms*, STOC 2002, 380-388]

Storing Documents

Simhash

1. Process documents into a set of features with associated weights (e.g., words with their frequencies).
2. Generate a hash value with b bits (desired size of fingerprint) for each word. The hash value should be unique for each word.
3. In a b-dimensional vector v, update the components of the vector by adding the weight for a feature (e.g., word) to every component for which the corresponding bit in the feature's hash value is 1, and subtracting the weight if the value is 0.
4. After all features (e.g., words) have been processed, generate a b-bit fingerprint by setting the ith bit to 1 iff the ith component of v is positive.

Storing Documents

Simhash example

Original text

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

Words with weights (stopwords removed)

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

8-bit hash values

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				

Vector v formed by summing the weights

1 -5 9 -9 3 1 3 3

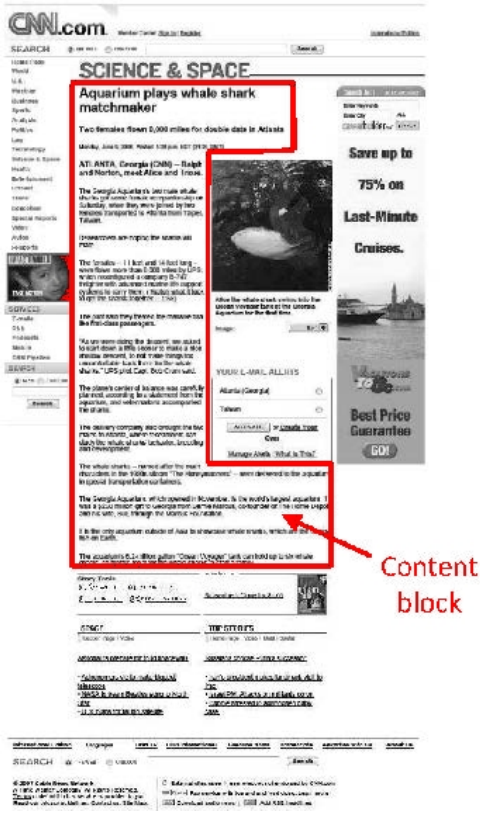
8-bit fingerprint formed from v

1 0 1 0 1 1 1 1

Storing Documents

Removing noise

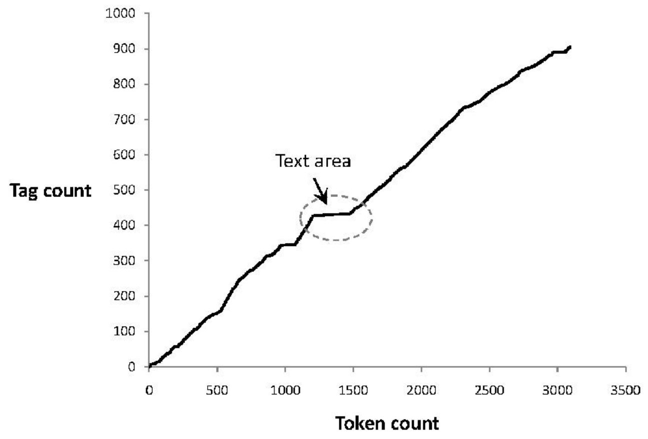
- ❑ Many web pages contain text, links, images that are not directly related to the main content of the pages
- ❑ This additional material is mostly noise that could negatively affect the ranking of the pages
- ❑ Techniques developed to detect the content blocks in a web page
  - – Non-content material then either ignored or reduced in importance during indexing


CNN.com
SCIENCE & SPACE
Aquarium plays whale shark matchmaker
Two females flown 8,000 miles for double date in Atlanta
Save up to
75% on
Last-Minute
Cruises.
Best Price
Guarantee
GO!
Content block

Storing Documents

First idea to find content blocks: document slope curve

- Cumulative distribution of tags in the example page



  – Main text content corresponds to the “plateau” in the middle
  – Represent a web page as a sequence of N bits, where $b_n = 1$ iff the nth 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
- This means to maximize

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

- Caution: Works only when proportion of text tokens lower than proportion of tags in non-content sections
- Better: Search low-slope sections of the curve

[Pinto et al., *QuASM: a system for question answering using semi-structured data*, JCDL 2002: 46–55]

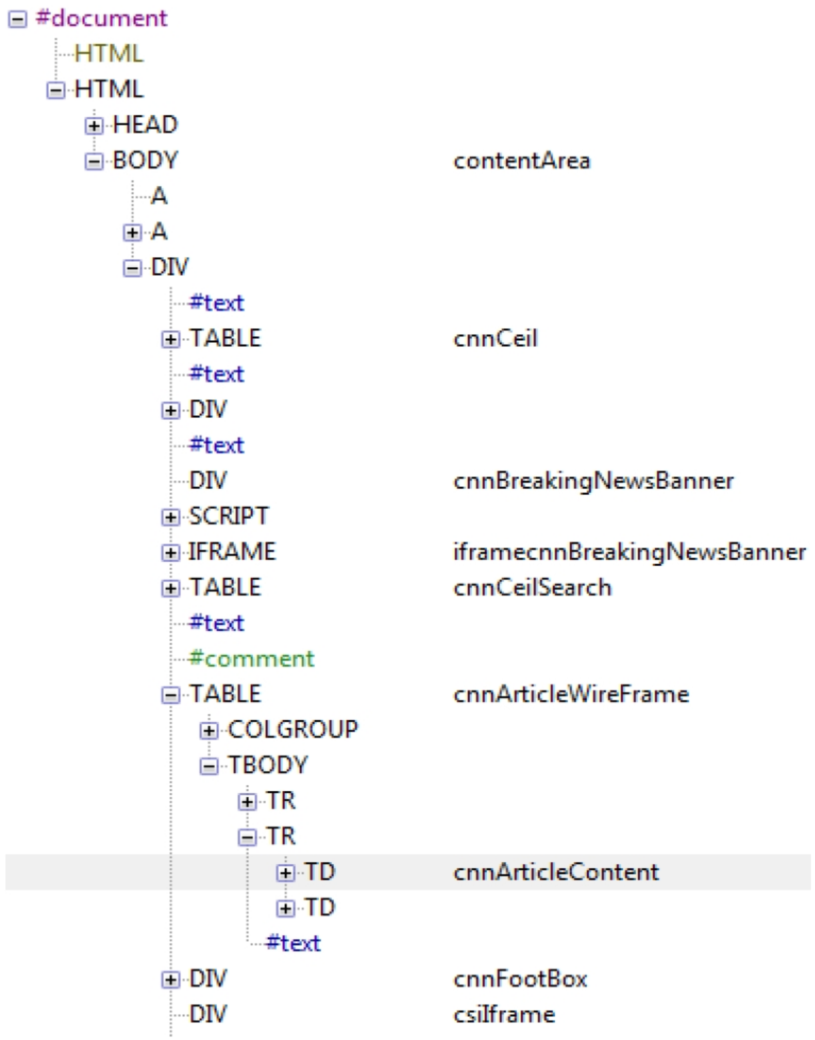
Storing Documents

Other ideas to find content blocks

- Use DOM structure and visual (layout) features
- Filtering techniques remove nodes of images, scripts, ads, link lists, and tables

  [Gupta et al., *DOM-based content extraction of HTML documents*, WWW 2003: 207–214]
- Position of a block, font size, background and font color, presence of separators (lines, spaces) also used

  [Yu et al., *Improving pseudo-relevance feedback in web information retrieval using web page segmentation*, WWW 2003: 11–18]



```
⊟ #document
    HTML
  ⊟ HTML
    ⊞ HEAD
    ⊟ BODY                        contentArea
        A
      ⊞ A
      ⊟ DIV
          #text
        ⊞ TABLE                   cnnCeil
          #text
        ⊞ DIV
          #text
          DIV                     cnnBreakingNewsBanner
        ⊞ SCRIPT
        ⊞ IFRAME                  iframecnnBreakingNewsBanner
        ⊞ TABLE                   cnnCeilSearch
          #text
          #comment
        ⊟ TABLE                   cnnArticleWireFrame
          ⊞ COLGROUP
          ⊟ TBODY
            ⊞ TR
            ⊟ TR
              ⊞ TD                cnnArticleContent
              ⊞ TD
              #text
        ⊞ DIV                     cnnFootBox
          DIV                     csiIframe
```