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
  - 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
      
      \[
      \begin{array}{cccccccc}
      T & r & o & p & i & c & a & l & f i s h & S u m \\
      54 & 72 & 6F & 70 & 69 & 63 & 61 & 6C & 20 & 66 & 69 & 73 & 68 & 508
      \end{array}
      \]
  - 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
Computing Similarity

- Features:
  - Segments of a document (natural or artificial breakpoints)
  - Shingles (Word N-Grams)
  - *a rose is a rose is a rose* → 4-grams are
    - a_rose_is_a
    - rose_is_a_rose
    - is_a_rose_is
    - a_rose_is_a
- Similarity Measure between two docs (= sets of shingles)
  - Jaccard coefficient: (Size_of_Intersection / Size_of_Union)

Shingles + Set Intersection

- Computing exact set intersection of shingles between all pairs of documents is expensive
- Approximate using a cleverly chosen subset of shingles from each (a sketch)
- Estimate (size_of_intersection / size_of_union) based on a short sketch
Sketch of a document

- Create a “sketch vector” (of size ~200) for each document
  - Documents that share ≥ t (say 80%) corresponding vector elements are deemed near duplicates
  - For doc $D$, sketch$_D[i]$ is as follows:
    - Let $f$ map all shingles in the universe to $1..2^m$ (e.g., $f$ = fingerprinting)
    - Let $\pi_i$ be a random permutation on $1..2^m$
    - Pick $\text{MIN} \{\pi_i(f(s))\}$ over all shingles $s$ in $D$

Computing Sketch[i] for Doc1

Document 1

- Start with 64-bit $f($shingles$)$
- Permute on the number line with $\pi_i$
- Pick the min value
Test if \( \text{Doc1.Sketch}[i] = \text{Doc2.Sketch}[i] \)

![Diagram showing sketches of Document 1 and Document 2 with points labeled A and B.](image)

Are these equal?

Test for 200 random permutations: \( \pi_1, \pi_2, \ldots, \pi_{200} \)

---

### Final notes

- **Shingling is a randomized algorithm**
  - Our analysis did not presume any probability model on the inputs
  - It will give us the right (wrong) answer with some probability on any input
- We’ve described how to detect near duplication in a pair of documents
- In “real life” we’ll have to concurrently look at many pairs
  - See text book for details
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

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.

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
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
<?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

- 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
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
<?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>
  </channel>
</rss>
```

RSS Example

```xml
...<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>
</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

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
  - does not uniquely specify bits to glyph mapping!
  - 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

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–127</td>
<td>0–7F</td>
<td>0xxxxxxx</td>
</tr>
<tr>
<td>128–2047</td>
<td>80–7FF</td>
<td>110xxxxx 10xxxxx</td>
</tr>
<tr>
<td>2048–55295</td>
<td>800–D7FF</td>
<td>1110xxxx 10xxxxx 10xxxxx</td>
</tr>
<tr>
<td>55296–57343</td>
<td>D800–DFFF</td>
<td>Undefined</td>
</tr>
<tr>
<td>57344–65535</td>
<td>E000–FFFF</td>
<td>1110xxxx 10xxxxx 10xxxxx</td>
</tr>
<tr>
<td>65536–1114111</td>
<td>10000–10FFFF</td>
<td>11110xxx 10xxxxx 10xxxxx 10xxxxx</td>
</tr>
</tbody>
</table>

– e.g., Greek letter pi ($\pi$) 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
  - 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
BigTable

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

![BigTable Diagram]

- 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
Size of the web

How big is the web?

• Issues
  – The web might as well be infinite
    • Dynamic content, e.g., calendar
  – Static web contains syntactic duplication, mostly due to mirroring (~30%)
  – Some servers are seldom connected

• Who cares?
  – Engine design
  – Media, and consequently the user
The web: size

- What is being measured?
  - Number of hosts
  - Number of (static) html pages
    - Volume of data
- Number of hosts – netcraft survey
  - Monthly report on how many web hosts & servers are out there
- Number of pages – numerous estimates (will discuss later)

---

**Netcraft Web Server Survey**

The web: evolution

- All of these numbers keep changing
- Relatively few scientific studies of the evolution of the web [Fetterly & al, 2003]
- Sometimes possible to extrapolate from small samples (fractal models) [Dill & al, 2001]

Rate of change

- [Cho00] 720K pages from 270 popular sites sampled daily from Feb 17 – Jun 14, 1999
  - Any changes: 40% weekly, 23% daily
- [Fett02] Massive study 151M pages checked over few months
  - Significant changed -- 7% weekly
  - Small changes – 25% weekly
- [Ntul04] 154 large sites re-crawled from scratch weekly
  - 8% new pages/week
  - 8% die
  - 5% new content
  - 25% new links/week
Static pages: rate of change

- Fetterly et al. study (2002): several views of data, 150 million pages over 11 weekly crawls
  - Bucketed into 85 groups by extent of change

![Diagram showing extent of change for different domains]

What can we attempt to measure?

- The relative sizes of search engines
  - The notion of a page being indexed is still *reasonably* well defined.
  - Already there are problems
    - Document extension: e.g. engines index pages not yet crawled, by indexing anchortext.
    - Document restriction: All engines restrict what is indexed (first $n$ words, only relevant words, etc.)

- The coverage of a search engine relative to another particular crawling process.
Statistical methods

- Random queries
- Random searches
- Random IP addresses
- Random walks

\[ A \cap B = \frac{1}{2} \times \text{Size A} \]
\[ A \cap B = \frac{1}{6} \times \text{Size B} \]

\[ \frac{1}{2} \times \text{Size A} = \frac{1}{6} \times \text{Size B} \]

\[ \therefore \frac{\text{Size A}}{\text{Size B}} = \frac{1}{6}/(1/2) = 1/3 \]

Sample URLs randomly from A
Check if contained in B
and vice versa

Each test involves: (i) **Sampling** (ii) **Checking**
Sampling URLs

• Ideal strategy: Generate a random URL and check for containment in each index.

• Problem: Random URLs are hard to find! Enough to generate a random URL contained in a given Engine.

Random URLs from random queries  [Bharat & B, 98]

• Generate random query: how?
  – Lexicon: 400,000+ words from a crawl of Yahoo!
  – Conjunctive Queries: \( w_1 \) and \( w_2 \)
    
      e.g., vocalists AND rsi

• Get 100 result URLs from the source engine
• Choose a random URL as the candidate to check for presence in other engines.
Query Based Checking

- **Strong Query** to check for a document \( D \):
  - Download document. Get list of words.
  - Use 8 low frequency words as AND query
- Check if \( D \) is present in result set.
- Problems:
  - Near duplicates
  - Frames
  - Redirects
  - Engine time-outs
  - Might be better to use e.g. 5 distinct conjunctive queries of 6 words each.

Advantages & disadvantages

- Statistically sound under the induced weight.
- Biases induced by random query
  - Query Bias: Favors content-rich pages in the language(s) of the lexicon
  - Ranking Bias: Solution: Use conjunctive queries & fetch all
  - Checking Bias: Duplicates, impoverished pages omitted
  - Document or query restriction bias: engine might not deal properly with 8 words conjunctive query
  - Malicious Bias: Sabotage by engine
  - Operational Problems: Time-outs, failures, engine inconsistencies, index modification.
Random searches

- Choose random searches extracted from a local log [Lawrence & Giles 97] or build “random searches” [Notess]
  - Use only queries with small results sets.
  - Count normalized URLs in result sets.
  - Use ratio statistics

Advantages & disadvantages

- Advantage
  - Might be a better reflection of the human perception of coverage

- Issues
  - Samples are correlated with source of log
  - Duplicates
  - Technical statistical problems (must have non-zero results, etc.)
Random searches
[Lawr98, Lawr99]

- 575 & 1050 queries from the NEC RI employee logs
- 6 Engines in 1998, 11 in 1999
- Implementation:
  - Restricted to queries with < 600 results in total
  - Counted URLs from each engine after verifying query match
  - Computed size ratio & overlap for individual queries
  - Estimated index size ratio & overlap by averaging over all queries

Queries from Lawrence and Giles study

- adaptive access control
- neighborhood preservation topographic
- hamiltonian structures
- right linear grammar
- pulse width modulation neural
- unbalanced prior probabilities
- ranked assignment method
- internet explorer favourites importing
- karvel thornber
- zili liu

- softmax activation function
- bose multidimensional system theory
- gamma mlp
- dvi2pdf
- john oliensis
- rieke spikes exploring neural
- video watermarking
- counterpropagation network
- fat shattering dimension
- abelson amorphous computing
Random IP addresses

[Lawrence & Giles ‘99]

- Generate random IP addresses
- Find a web server at the given address
  - If there’s one
- Collect all pages from server.
  
  http://digitalarchive.oclc.org/da/ViewObject.jsp?objid=000003447

Random IP addresses [ONei97, Lawr99]

- HTTP requests to random IP addresses
  - Ignored: empty or authorization required or excluded
  - [Lawr99] Estimated 2.8 million IP addresses running crawlable web servers (16 million total) from observing 2500 servers.
  - OCLC using IP sampling found 8.7 M hosts in 2001
    - Netcraft [Netc02] accessed 37.2 million hosts in July 2002
  - [Lawr99] exhaustively crawled 2500 servers. Estimated size of the web to be 800 million
    - Estimated use of metadata descriptors:
      - Meta tags (keywords, description) in 34% of home pages, Dublin core metadata in 0.3%
Advantages & disadvantages

• Advantages
  – Clean statistics
  – Independent of crawling strategies
• Disadvantages
  – Doesn’t deal with duplication
  – Many hosts might share one IP, or not accept requests
  – No guarantee all pages are linked to root page.
    • Eg: employee pages
  – Power law for # pages/hosts generates bias towards sites with few pages.
    • But bias can be accurately quantified IF underlying distribution understood
  – Potentially influenced by spamming (multiple IP’s for same server to avoid IP block)

Conclusions

• No sampling solution is perfect.
• Lots of new ideas ...
• ....but the problem is getting harder
• Quantitative studies are fascinating and a good research problem