CSCI 599: Applications of Natural Language Processing – Information Retrieval

Crawls and Feeds
Indexing Process

- E-mail, Web pages, News articles, Memos, Letters
- Text Acquisition
- Text Transformation
- Document data store
- Index Creation
- Index
Indexing Process

- **Text acquisition**
  - identifies and stores documents for indexing

- **Text transformation**
  - transforms documents into *index terms or features*

- **Index creation**
  - takes index terms and creates data structures (*indexes*) to support fast searching
Web Crawler

- Finds and downloads web pages automatically
  - provides the collection for searching
- Web is huge and constantly growing
- Web is not under the control of search engine providers
- Web pages are constantly changing
- Crawlers also used for other types of data
Retrieving Web Pages

- Every page has a unique \textit{uniform resource locator} (URL)
- Web pages are stored on web servers that use HTTP to exchange information with client software
- e.g.,

\begin{verbatim}
http://www.cs.umass.edu/csinfo/people.html
\end{verbatim}

\begin{verbatim}
scheme  hostname  resource
http    www.cs.umass.edu  /csinfo/people.html
\end{verbatim}
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.
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 the 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
procedure CRAWLER_THREAD(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
    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

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

$$\text{Age}(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx$$
- 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
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
<?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
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>
  </channel>
</rss>
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
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 10xxxxxx</td>
</tr>
<tr>
<td>2048–55295</td>
<td>800–D7FF</td>
<td>1110xxxx 10xxxxxx 10xxxxxx</td>
</tr>
<tr>
<td>55296–57343</td>
<td>D800–DFFF</td>
<td>Undefined</td>
</tr>
<tr>
<td>57344–65535</td>
<td>E000–FFFFF</td>
<td>1110xxxx 10xxxxxx 10xxxxxx</td>
</tr>
<tr>
<td>65536–1114111</td>
<td>10000–10FFFFF</td>
<td>11110xxx 10xxxxxx 10xxxxxx 10xxxxxx</td>
</tr>
</tbody>
</table>

- 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
  - 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
- 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