Approaches to Mining the Web

Olfa Nasraoui
University of Louisville
Web Mining: Mining Web Data (3 Types)

- Structure Mining: extracting info from topology of the Web (links among pages)
  - Hubs: pages *pointing to* many other pages
  - Authorities: pages *pointed to by* many other pages
  - Communities: pages forming islands
- Usage Mining: extracting info on how *people* visit the above pages, ex: which route through site leads to checkout?
- Content Mining: Extracting useful info from page *content* (text, images, other...etc),
  - used by search engines, agents, recommendation engines to help users find what they are looking for
The Web from the perspective of data mining

- WWW = pages connected by links
- A Web server provides access to the pages on a website
- A Web page may consist of several components/frames
- A pageview $\leftrightarrow$ several frames
- From now on, page $\leftrightarrow$ 1 frame
Idealized Data Representation

- **Structure**: Directed graph (nodes: pages, edges: links)
- **Content**: Index:
  - string/word/phrase → page
- **Usage**: Profile: Each profile summarizes sites, paths, queries, documents read, items purchased, …etc

- relatively static/rigid compared to usage
- Not static: behavior of users OVER TIME
What can each type of mining do?

• Structure: organize, cluster, rank by authority, …etc

• Content: Information retrieval

• Usage: personalization, Info retrieval
  – Short time frame: single session
  – Long time frame: repeat sessions/repeat visitors (need ID, ex: registered customer)
<table>
<thead>
<tr>
<th>Type of data</th>
<th>Ideal data</th>
<th>Real life</th>
<th>Applications</th>
<th>Future trends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content: (Web crawlers collect this data, they can have global view)</td>
<td>-Complete -Up-to-date</td>
<td>-Easy to obtain -Public -Unrestricted</td>
<td>Information retrieval</td>
<td></td>
</tr>
<tr>
<td>Structure: (Web crawlers collect this data, they can have global view)</td>
<td>-Complete -Up-to-date</td>
<td>-Easy to obtain -Public -Unrestricted</td>
<td>Information retrieval, social networks</td>
<td></td>
</tr>
<tr>
<td>Usage: Web logs + App server logs + databases (external data: product, transaction, etc)</td>
<td>-Complete -Up-to-date</td>
<td>-Hard to obtain -Private -Restricted -Scattered</td>
<td>User profiling, personalization</td>
<td>Information retrieval</td>
</tr>
</tbody>
</table>
Part 1: Mining Structure: **Global Structure**

- Links reveal many things:
  - Hubs & authorities $\Rightarrow$ popularity
  - Islands $\Rightarrow$ can help disambiguate synonyms & organize by topic/community

- Web: directed graph
  - Link analysis: branch of DM concerned w/ finding patterns in graphs & networks
Global Structure … continued

- Counting citations: # of citations to an article = main evidence of its usefulness
- The more links leading to a site → the more important it must be
- Yet, an accurate picture of Web links is difficult because the Web is not static.
  - *Static* link: fixed URL
  - *Dynamic* link: not fixed, ex: generated as response to a search/query.
Kleinberg’s algorithm: Hubs & Authorities

- Hubs & authorities: measure of usefulness of a page
- Popular sites such as yahoo! Will be ranked higher than less popular but more authoritative subject-specific pages (because of the # links)
- Solution: rank not by total # of links, but # of subject-related links pointing to them
- Essential feature: pages based not only on content (root set), but also on link analysis: larger pool contains pages w/ in-links/out-links relative to the root set
- Larger pool contains more global structure ➔ can be mined to determine authoritative pages
Kleinberg’s Algorithm

- Phase 1: Create root set
- Phase 2: Identify candidate set
- Phase 3: Rank Hubs and authorities
Phase 1: Creating the root set
(Based on CONTENT ONLY!)

• Preprocess search string:
  – Eliminate stop list words (the, a, for, …etc)
  – Stemming: reduce words to base root (going → go, referral → refer, dogs → dog, … etc)

• Search the Web index consisting of mapping between {words → pages}

• Score each page based on terms it contains (TF.IDF):
  – Term rarity in entire collection of pages (IDF)
  – Term frequency within this page (TF)

• Sort and select top $n$ (typically $n = 200$)

• NOTE: we will discuss TF, IDF when we talk about content mining or text mining
Phase 2: Identifying candidate set (content enriched w/ structure)

• Expand root set by including:
  – Pages w/ links to each page in root set
  – Pages w/ links from each page in root set

• Limit # of pages brought into candidate set by any single member of root set to parameter ($d$, typically $d = 50$)

• Typical size of candidate set = 1000 – 5000

• Possible refinement strategies:
  – Filter out any links from within same website/domain as a page in root set (most purely navigational)
  – Diversification: remove bias/redundancy by limiting # of pages from same website brought into candidate set by any single member of root set to parameter ($m$)
Phase 3: Ranking Hubs and authorities

- Divide candidate set into hubs & authorities
- Main assumption: hubs & authorities have mutually reinforcing relationship:
  - A strong *hub* links to many authorities
  - A strong *authority* is linked to by many hubs
- HITS algorithm (*Hyperlink-Induced Topic Search*):
  - For each page $p$, a value $a_p$ in $[0,1]$, measures its *authority*
  - For each page $p$, a value $h_p$ in $[0,1]$, measures its *strength as a hub*
HITS Algorithm  
(Hyperlink-Induced Topic Search)

• If page is pointed to by many good hubs, we update the value of $a_p$ for the page $p$, to be the sum of $h_q$ over all pages $q$ that link to $p$:

  $$a_p = \sum h_q, \text{ sum over all } q \text{ such that } q \rightarrow p$$

• If a page points to many good authorities, we increase its hub weight:

  $$h_p = \sum a_q, \text{ sum over all } q \text{ such that } p \rightarrow q.$$
HITS Algorithm
(Hyperlink-Induced Topic Search)

Initialize \( a_p \) and \( h_p \) for all pages \( p \) in candidate set;

REPEAT {
   \( a_p = \sum h_q \), sum over all \( q \) such that \( q \rightarrow p \);
   Normalize \( a_p \) by dividing by sum of squares;
   \( h_p = \sum a_q \), sum over all \( q \) such that \( p \rightarrow q \);
   Normalize \( h_p \) by dividing by sum of squares;
}
UNTIL Convergence
HITS Algorithm  (continued…)

• Systems based on the HITS algorithm:
  – *Google*: achieve better quality search results than those generated by purely content based {term-index} engines such as AltaVista and those based on *directories* created by human ontologists such as Yahoo!

• Difficulties from ignoring textual contexts:
  – Topic Drift: when a hub contain multiple topics to *unrelated* subjects
  – Topic hijacking: when many pages from a single Web site point to the same single *popular* site (even when unrelated to subject of search!)
Local Structure

• **Sticky vs. slippery** page on same website
• Stickiness: metric that measures how long visitors stay on same site
• **Sticky Pages:** should be *Destination* pages: Content! (information, products…etc)
• **Slippery pages:** opposite of above: exist only to provide links to other pages: *Navigational Pages*
  – In a way checkout pages should not be sticky or this may indicate a problem!
• Local structure of website largely depends on goals
Part 2: Mining Usage Patterns

- Even when main concern is content or structure, usage may be helpful!
- Ex: can measure popularity of a page by # of visitors
- *Clickstream*: lowest level data for web usage mining = series of requests for pages received by web server that hosts a site
- *Hits*: Record of every GIF, JPEG, and HTML file requested by the user’s browser
- These hits can be aggregated into *page views*
- Page views can be further aggregated into *sessions*
- Can also link *several* sessions to an identified *visitor* for more effective *market basket analysis*
Different Data Sources

- **Web Logs: Low level**
  - Tracks individual items (even those that are embedded within a web page like graphics, ...etc) requested by a Web browser

- **Application logs: Higher level**
  - In e-commerce architecture, page content constructed on the fly by an application server
  - Same URL (in web log) may correspond to different web content (logged on app server)
  - App servers know when customers check in and check out, items placed or removed from shopping cart, ...etc
Web Logs

- Web Logs: Low level data
  - Tracks individual items (even those that are embedded within a web page like graphics, etc) requested by a Web browser (see next page)
  - Often several servers involved: ad server (banners), web servers (text, images), app. Servers (content calculated on fly, ex: shopping cart data w/ prices etc)
  - Hence: user clicks on 1 link turns into multiple hits in website these hits may be recorded across multiple logs on different servers...
Excerpt from web log file

**Format:** time stamp, originating IP address, method, requested URL, error status code

<table>
<thead>
<tr>
<th>Time Stamp</th>
<th>IP Address</th>
<th>Method</th>
<th>Requested URL</th>
<th>Status Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:11:02</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/halford/halford.html</td>
<td>200</td>
</tr>
<tr>
<td>17:11:21</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/griffin/griffin.html</td>
<td>200</td>
</tr>
<tr>
<td>17:11:25</td>
<td>141.225.160.198</td>
<td>GET</td>
<td>/graphics/schematics.jpg</td>
<td>304</td>
</tr>
<tr>
<td>17:11:48</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/graphics/griffin.jpg</td>
<td>200</td>
</tr>
<tr>
<td>17:11:48</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/nasraoui/nasraoui.html</td>
<td>200</td>
</tr>
<tr>
<td>17:11:48</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/nasraoui/frame_content.html</td>
<td>200</td>
</tr>
<tr>
<td>17:11:48</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/nasraoui/home1.html</td>
<td>200</td>
</tr>
<tr>
<td>17:11:48</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/nasraoui/GRAPHICS/BACKGROUNDS/ Horizon.jpg</td>
<td>200</td>
</tr>
<tr>
<td>17:11:48</td>
<td>141.225.195.29</td>
<td>GET</td>
<td>/people/faculty/nasraoui/GRAPHICS/smallmail.gif</td>
<td>200</td>
</tr>
</tbody>
</table>

**Additional data possible:** referrer address (the last page where the user was), size of data transferred, protocol (mainly HTTP on a web server), user agent (browser type)
Web Log Data Preparation

• **Step 1: Filtering**: remove all log entries representing requests for graphic files (jpeg, gif, png, …, etc.)

• **Step 2: Despidering**:
  - removing entries generated by requests from spiders/crawlers (ex: used by search engines to index Web pages) and other bots (ex: performance monitoring systems)
  - Some spiders can be recognized by name (in agent field of log entry), also by their access patterns (heuristically)
Web Log Data Preparation... continued

- **Step 3: User identification**: a prerequisite to sessionization:
  - Identify requests made by same user during *single* visit
    - User registration/login (only sure way)
    - Alternatives based on heuristics: Combination of IP Address + Agent field (browser)
  - Identify requests made by same user during *multiple* visits
    - Return visitors are recognized either by registration mechanisms or by cookies
    - Cookies: text strings issued by web server and stored by the browser on client computer. They uniquely identify a user session. In future, Web server requests o see if any of its past cookies is stored on user’s computer to recognize user
    - Assumptions: user has “accepted” to store cookie in old session, has not disabled or deleted cookies in new session, and is using same computer + browser
• **Step 4: Sessionization:**
  – identifying a series of pageviews as requested by *same* user during a single *visit*
  – Heuristics: all requests from same user within a maximal time duration between consecutive requests
  – Same user: recognized
    • by cookie, or
    • by modifying all URLs on currently requested page to include unique session identifier ➔ If user clicks on any link from this page, the web server will be able to track the session by simply parsing the requested URL to extract the user ID
  – Web logs may be insufficient: hard to estimate *how long* a user was viewing a page
Web Log Data Preparation … continued

• **Step 5: Path completion:** Web logs may be insufficient:
  
  – caching may hide some requests:
    
    • Browser keeps a cache of recently visited pages for days. When user requests this page, it is loaded locally without any request to server (unless “reload” page is selected)
    
    • Caching can also occur at a different level: proxy server (handles requests in an intermediary fashion between the user and the destination server (ex: corporate proxy or Internet Service Provider proxy)
    
  – Another proxy-caused problem: For different users that connect via the same proxy, IP address of proxy is the one that is logged in the Web log ➔ difficult to distinguish between different users
Web Log Data Preparation … continued

• Path completion: Some solutions:
  • Using referrer info (+ site structure) can approximately reconstruct a session
  • Referrer info can be saved in Web log: records where user was (previous page) just before user requested a local page

• Good News for the caching problem:
  – Dynamic web pages (created using Java Server Pages (JSP) or Microsoft’s Active Server Pages (ASP) are not cached because each dynamically created page is unique.
Application logs

• Higher level type of data
  – In e-commerce architecture, page content constructed on the fly by an application server
  – Same URL (in Web log) may correspond to different web content (logged on app. server)
  – Example: request for the price of a product may return different prices depending on the date, promotions, …etc
  – App servers know when customers check in and check out, when items are placed or removed from shopping cart, …etc.
  – Will discuss more in next chapter…
Application: Usage Mining to Improve Site Usability

- Very common application of Web usage mining
- Treat user sessions consisting of page requests like market baskets consisting of items purchased
- Analyze data using association rule mining
  - Discovered associations (A => B) may suggest additional links between unconnected pages
  - This makes site easier to navigate, and allows users to find what they need faster…
- Analyze data to group similar sessions into clusters
  - Clusters may correspond to user profiles / modes of usage of the website
  - Knowing user profiles may suggest improvements to web site design and even dynamic website design during same session
Web Usage Mining Tasks: Associations & Sequences

- Treat user sessions consisting of page requests like market baskets consisting of items purchased

- Analyze data using association rule mining
  - Discovered associations (A \Rightarrow B) may suggest additional links between unconnected pages
  - This makes site easier to navigate, and allows users to find what they need faster...
  - Suggest recommendations for associated products based on current basket (cross-sell)

- Sequential pattern discovery: extension of association rules mining that discovers patterns of co-occurrence incorporating the notion of time sequence.
  - Pattern: a Web page or a set of pages accessed immediately after another set of pages.
  - \Rightarrow help to discover users’ trends, and make predictions about visit patterns
Web Usage Mining Tasks: Clustering, Collaborative Filtering

• Clustering: Analyze data to group similar sessions into clusters
  – Clusters may correspond to user profiles / modes of usage of the website
  – Knowing user profiles may suggest
    • improvements to web site design
    • dynamic website design during same session
    • Design of several smaller websites or stores

• Collaborative Filtering
  – Recommend products on a site with large # of products
  – Lazy learning: no need for learning, based on examples of past usage
  – Problems: sparse data: many more products/pages than customers/browsers, scaling up to huge # of visitors & # of items or pages
Example: Effect of Recommendation System on a home shopping group

<table>
<thead>
<tr>
<th></th>
<th>With traditional methods</th>
<th>With cross-sell real-time recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Cross-Sell Value</td>
<td>$19.50</td>
<td>60% higher</td>
</tr>
<tr>
<td>Cross Sell Success Rate</td>
<td>9.8%</td>
<td>50% higher</td>
</tr>
</tbody>
</table>

Source: J. Riedl, “Why Does KDD Care About Personalization?”
Mining Web Content

• Task most popularly performed by Web search engines
• Information Retrieval (IR): formal name for task performed by search engine to help users find what they look for.
• Web Content:
  – HTML: Hypertext Markup Language: standard for describing the way document should be displayed using tags.
  – XML: eXtensible Markup Language: extensible standard that allows community of users to agree on application-specific set of tags that help in understanding what document info means!
Content Mining ... continued

• Most current content mining ↔ Text mining
• Information retrieval:
  – Find what user is looking for
  – But without finding too much else!!!
  – These two conflicting ideas are captured by 2 measures of IR:
    • **Precision**: Of the pages returned, what proportion are correct?
    • **Recall**: Of all the pages that are correct, what proportion are returned?
    • Typically
      – Perfect recall ➔ very low precision
      – Perfect precision ➔ very low recall
Content based Classification

• Classification
  – Very common application of data mining
  – Assign a class label to a data record

• In Web content mining:
  – Assign keywords to a Web page (to predict its topic)
  – Assign a language to a Web page (ex: to restrict search results to a given language)
  – Challenge: Most data mining methods for classification work w/ structured data
    • However free text is unstructured!
    • Traditionally, free text was transformed into structured data (in the form of keyword vectors) via feature (i.e. keyword) extraction
Automatic Classification of Web Documents

• Assign a class label to each document from a set of predefined topic categories
• Based on a set of examples of pre-classified documents
• Example:
  – Use Yahoo!'s taxonomy and its associated documents as training and test sets
  – Derive a Web document classification scheme
  – Use the scheme to classify new Web documents
• Keyword-based document classification methods (mostly using vector space model of documents as a vector of keyword/term frequencies…etc)
• Statistical models (ex: Naïve Bayes: models document classes as word probability distributions)
More Content based Classification Methods

• Genetic Algorithms can be used to evolve classifier of text documents
  – Ex: classify customer comments/e-mails/web blogs as positive or negative?

• Memory based reasoning or K-Nearest neighbors based classification (K-NN):
  – assign **keywords** to a web page
  – Compare a new page to a set of pre-classified examples
  – The *k* most **similar** pages are used to classify the new page using **weighted voting**
  – Confidence in classification: related to degree of similarity to examples…
Vector Space Model

- **Vector Space Model**: documents represented using *keywords* in text documents.
- Documents are vectors in this s-dimensional space.
- Mathematical model: $m \times s$ matrix where each row $\Leftrightarrow$ 1 document, each column $\Leftrightarrow$ 1 term or keyword

$$
\begin{array}{c|cccc}
 & t_1 & t_2 & \cdots & t_s \\
\hline
 d_1 & 0 & 1 & \quad & 0 \\
 d_2 & 0 & 2 & \quad & 3 \\
 \cdots & \quad & \quad & \quad & \quad \\
 d_m & 2 & 0 & \quad & 1 \\
\end{array}
$$

- Entries in matrix: If keyword $\# j$ occurs $n$ times in document $\# i$, then the content of [row $i$, col $j$] = $n$
Document Vector

- $D=\{d_1,...,d_n\}$ of documents and $T=\{t_1,...,t_s\}$ of indexing/querying terms.
  - **Vector Space model**: each document is represented as a vector of dimension $s = \text{the number of terms}$.
    
    $$d_i = \langle w_{i1}, w_{i2}, \ldots, w_{is} \rangle$$

- $w_{ij}$ is weight of term $t_j$ in $d_i$.
- Let $f_{ij} = \text{freq of occurrence of term } t_j \text{ in } d_i$
  
  $$w_{ij} = f_{ij} \cdot \log\left[\frac{N}{N_j}\right]$$

  - $N = \# \text{ of documents}$
  - $N_j = \# \text{ of documents in which term } t_j \text{ occurs at least once}$
  - Inverted Document Frequency $(IDF) = \log (N/N_j)$ measures rarity of term
  - $w_{ij}$ typically normalized: divide by $\sum w_{ij}, \sum w_{ij}^2$, or max $\{w_{ij}\}$. 
Similarity

- Assessing similarity between two Web pages
- *Similarity* is inversely related to *distance*
- Euclidean distance is *not* appropriate for text data (huge dimensionality, asymmetrical attributes/words)
- Intuitively, similarity between two pages should depend on the # of words in common compared to total # of words in both pages
- Ex: Cosine similarity = \#common words / 
  \((\#\text{words in page 1} \times \#\text{words in page 2})\)

\[
SIM(d_i, d_q) = \frac{\sum_{j=1}^{s} w_{ij} * w_{qj}}{\sqrt{\sum_{j=1}^{s} w_{ij}^2} \sqrt{\sum_{j=1}^{s} w_{qj}^2}}
\]
Case Study: Classification of Customer Comments as Compliment or Complaint

• Business context: customers comment on quality of service of airline company: e-mails, fill comment forms on website,…etc

• Data: free text + other (ex: flight #, service class, date, destination, …etc)

• Preprocessing:
  – Free text → document vector + Add metadata about comment (size in bytes, # distinct words)

• Data Mining Task: Classification (class 0: compliment, class 1: complaint)
Case Study: Classification of Customer Comments as Compliment or Complaint

- Data Mining Task: Classification (class 0: compliment, class 1: complaint)
- Genetic Programming is used to evolve prediction tree
- Solutions similar to single layer Neural Net
- Chromosome encodes a solution:
  - each gene = attribute or operation
- Chromosomes compete based on fitness
- 85% classification accuracy
  - Enough to speed up complaint processing!
- Important attributes:
  - comment length → complaint
  - Some words: baggage, courteous, experience
Clustering Web Documents to Support Searching

- Search engine can help users find what they need by clustering search results into folders and sub-folders.
- *Northern Light* organizes the Web into custom search folders
  - Folders are labeled by humans (hence semi-automatic)
  - Accessible only by registration / subscription
  - Newest version: Enterprise Search Engine (to be used inside a business ➔ Knowledge Management)
- Example: *Vivissimo* clusters search results on the fly...without any labeling (see next slide...)
- Difference is in the use of a *concept hierarchy*!
- Most extreme use of such a hierarchy: *yahoo!* (entirely manual)
Cluster "Discovery, Knowledge"
Cluster “Conference”

1. **KDD-2001: The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**
   - Chairs: PC KDD-2001 The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
   - August 26 - 29, 2001 San Francisco, California, USA
   - ACM Special Interest Group on Knowledge Discovery
   - URL: [www.acm.org/sigkdd/kdd2001 - show in clusters](http://www.acm.org/sigkdd/kdd2001)
   - Sources: [Wisent2](http://www.wisent2.com)

2. **SIAM International Conference on Data Mining**
   - URL: [www.siam.org/meetings/sdm02 - show in cluster](http://www.siam.org/meetings/sdm02)
   - Sources: [Wisent13, MCH 29](http://www.wisent13.com)

3. **ICDM-03: 2003 IEEE International Conference on Data Mining**
   - Your browser does not support frames. Please see non-frames version.
   - URL: [www.cs.uvm.edu/~xwu/icdm-03.html - show in cluster](http://www.cs.uvm.edu/~xwu/icdm-03.html)
   - Sources: [Wisent16, Open Directory 45, MCH 86](http://www.wisent16.com)

4. **SAS - M2004 Data Mining Technology Conference**
   - [www.sas.com/events/m2004 - show in cluster](http://www.sas.com/events/m2004)
   - Source: [Wisent16](http://www.wisent16.com)
Search query = “mining”

Discovered Clusters
Extracting Information from Free Text

- Feature/Information extraction: important to extract specific data, ex: price, product attributes, vendors, ... etc
- Useful in e-commerce: comparison shopping
- This has been made easier by recent structuring of content using XML tags
Summary of Approaches for Mining the Web

- **Structure Mining:**
  - analyzing links between pages
  - *Global vs. local* structure
    - Global structure ➔ identify *hubs, authorities*
    - Local structure ➔ understand site’s intended purpose, detect design problems
  - Typically based on graph analysis
  - HITS algorithm
  - App: search engines (for global structure)

- **Usage Mining:**
  - Analyzing user behavior *over time*
  - Build user *profiles* of *anonymous* visitors
  - Data mining tasks: association rule discovery, clustering
  - App: personalization, marketing, etc

- **Content Mining**
  - Extract info from web pages
  - Data mining tasks: clustering, classification
  - App: search engines