Web Mining

Web Usage Mining

Prof. Dr. Christian Bizer
Oliver Lehmberg

FSS 2016
Web Usage Mining

Definition

Discovery of patterns in click-streams and associated data collected or generated as a result of user interactions with one or more Web sites.

Typical Sources of Data

1. web server access logs
2. e-commerce and product-oriented user events (e.g., shopping cart changes, ad or product click-throughs, purchases)
3. user events on social network sites (e.g., likes, posts, comments)
4. user profiles and/or user ratings
5. page attributes, page content, site structure
6. additional domain knowledge and demographic data
The Web Usage Mining Process

Data Preparation Phase

- Web & Application Server Logs
  - Data Cleaning
  - Pageview Identification
  - Sessionization
  - Data Integration
  - Data Transformation

Pattern Discovery Phase

- Aggregate User models
- Pattern Analysis
- Pattern Filtering
- Aggregation
- Characterization
- Patterns

Usage Mining

- Transaction Clustering
- Pageview Clustering
- Correlation Analysis
- Association Rule Mining
- Sequential Pattern Mining
Chapter Outline

1. Usage Data Collection

2. Usage Data Preprocessing
   1. User and Session Identification
   2. Data Aggregation and Semantic Enrichment

3. Usage Mining Tasks

4. Recommender Systems
   1. Collaborative Filtering
   2. Content-based Recommendation
   3. Hybrid Recommendation Approaches
   4. Evaluating Recommender Systems
   5. Attacks on Recommender Systems
Literature

- Bing Lui: Web Data Mining. Chapter 12: Web Usage Mining
- Jannach et al.: Recommender Systems. Chapters 2, 3, 5, 7
- Lots of thanks to Bing Lui, Jannach et al., and Bettina Berndt for the original versions of the slides.
1. Usage Data Collection

- **Server-Side Data Collection**
  - **Traditional Web Server Logs**
    - Content: IP, timestamp, page, browser, …
    - Format: text files, database
  - **Application Logs**
    - Specific application events (e.g. change in shopping basket)
  - Restricted to single server

- **Client-Side Data Collection**
  - via Page Tagging or Toolbars
  - Additional collectable data:
    - mouse movements
    - keyboard strokes
    - size of browser window
  - Not restricted to single server
Recording Users Entering and Leaving the Site

Logs may extend beyond visits to the site and show:

- where a visitor was before (via HTTP referer)

```
```

- and where she went next (via URL rewriting):

![Yahoo Search Result](image)
Examples of Large Usage Data Collections

Enable the
• analysis of the current interests and behavior of the world’s population.
• identification of suspected terrorists.
Content of a typical Apache web server log:

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Date/Time</th>
<th>Method</th>
<th>File Path</th>
<th>Status Code</th>
<th>Bytes</th>
<th>Referrer</th>
<th>User Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>203.252.234.33</td>
<td>[01/Jun/1999:03:12:31]</td>
<td>GET</td>
<td>/</td>
<td>200</td>
<td>4980</td>
<td>&quot;&quot;</td>
<td>&quot;Mozilla/4.06 [en] (Win95; I)&quot;</td>
</tr>
</tbody>
</table>
Data Preprocessing

1. Data Cleansing
   - remove irrelevant log entries and fields in server logs
     - usually: remove all log entries related to images or scripts
     - ignoring certain page-views / items
   - remove log entries due to spider navigation

2. Data Integration
   - synchronize data from multiple server logs
   - integrate semantics, e.g. meta-data (e.g., content labels), e-commerce and application server data, registration data

3. Data Transformation
   - user identification
   - session identification
   - data aggregation

4. Data Reduction
   - sampling
Example: Web Crawler Traffic

- Saturday
- Sunday
- Monday
- Tuesday
- Wednesday
- Thursday
- Friday

HUMAN-GENERATED TRAFFIC

ROBOT-GENERATED TRAFFIC

Time (in hours)
# Mechanisms for User Identification

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Privacy Concerns</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address + Agent</td>
<td>Assume each unique IP address/Agent pair is a unique user</td>
<td>Low</td>
<td>Always available. No additional technology required.</td>
<td>Not guaranteed to be unique. Defeated by rotating IPs.</td>
</tr>
<tr>
<td>Embedded Session Ids</td>
<td>Use dynamically generated pages to associate ID with every hyperlink</td>
<td>Low to medium</td>
<td>Always available. Independent of IP addresses.</td>
<td>Cannot capture repeat visitors. Additional overhead for dynamic pages.</td>
</tr>
<tr>
<td>Registration</td>
<td>User explicitly logs in to the site.</td>
<td>Medium</td>
<td>Can track individuals not just browsers</td>
<td>Many users won't register. Not available before registration.</td>
</tr>
<tr>
<td>Cookie</td>
<td>Save ID on the client machine.</td>
<td>Medium to high</td>
<td>Can track repeat visits from same browser.</td>
<td>Can be turned off by users.</td>
</tr>
<tr>
<td>Software Agents</td>
<td>Program loaded into browser and sends back usage data.</td>
<td>High</td>
<td>Accurate usage data for a single site.</td>
<td>Likely to be rejected by users.</td>
</tr>
</tbody>
</table>

Examples of Agents: Page Tags (use javascript), Browser Plugins

Not anymore.
**Sessionization Heuristics**

*Time oriented heuristics*

- **h1**: Total session duration must not exceed a maximum
  - Threshold: 30 minutes

- **h2**: Page stay times must not exceed a maximum
  - Threshold: 10 minutes

*Navigation oriented heuristic*

- **href**: A page must have been reached from a previous page in the same session - except if the referrer is undefined, and the time elapsed since the last request is below 10 seconds

Source: Spiliopoulou et al., 2003
Data Aggregation

- aggregate log data in order to generate features that are suitable for the chosen task and mining algorithm.

Examples of possible Features

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalPages</td>
<td>Total number of pages retrieved in a Web session</td>
</tr>
<tr>
<td>ImagePages</td>
<td>Total number of image pages retrieved in a Web session</td>
</tr>
<tr>
<td>TotalTime</td>
<td>Total amount of time spent by Web site visitor</td>
</tr>
<tr>
<td>RepeatedAccess</td>
<td>The same page requested more than once in a Web session</td>
</tr>
<tr>
<td>ErrorRequest</td>
<td>Errors in requesting for Web pages</td>
</tr>
<tr>
<td>GET</td>
<td>Percentage of requests made using GET method</td>
</tr>
<tr>
<td>POST</td>
<td>Percentage of requests made using POST method</td>
</tr>
<tr>
<td>HEAD</td>
<td>Percentage of requests made using HEAD method</td>
</tr>
<tr>
<td>Breadth</td>
<td>Breadth of Web traversal</td>
</tr>
<tr>
<td>Depth</td>
<td>Depth of Web traversal</td>
</tr>
<tr>
<td>MultiIP</td>
<td>Session with multiple IP addresses</td>
</tr>
<tr>
<td>MultiAgent</td>
<td>Session with multiple user agents</td>
</tr>
</tbody>
</table>

This features could be used for robot detection (classification)
Data Aggregation

- **Example of a User Pageview Matrix**

  ![User Pageview Matrix](image)

- **Useful for discovering user groups (clustering)**
Semantic Enrichment

Basic Idea

Associate each requested page with one or more domain concepts, to better understand user behavior.

The request for a page signals interest in the concept(s).

Levels of the Analysis

- Page Level: 1 request ➔ 1 concept or n concepts
- Session Level: set / sequence of pages ➔ 1 concept or n concepts

Concepts can be part of a concept hierarchy or ontology

Useful for building/maintaining user profiles
Example: Semantic Enrichment

- **Input:** User Pageview Matrix

<table>
<thead>
<tr>
<th></th>
<th>A.html</th>
<th>B.html</th>
<th>C.html</th>
<th>D.html</th>
<th>E.html</th>
</tr>
</thead>
<tbody>
<tr>
<td>user1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>user2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>user3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>user4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>user5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>user6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Input:** Page Topic Matrix

<table>
<thead>
<tr>
<th></th>
<th>A.html</th>
<th>B.html</th>
<th>C.html</th>
<th>D.html</th>
<th>E.html</th>
</tr>
</thead>
<tbody>
<tr>
<td>web</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>mining</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>business</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>intelligence</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>marketing</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ecommerce</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>search</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>information</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>retrieval</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Result:** User Topic Matrix

<table>
<thead>
<tr>
<th></th>
<th>web</th>
<th>data</th>
<th>mining</th>
<th>business</th>
<th>intelligence</th>
<th>marketing</th>
<th>ecommerce</th>
<th>search</th>
<th>information</th>
<th>retrieval</th>
</tr>
</thead>
<tbody>
<tr>
<td>user1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>user2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>user3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>user4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>user5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>user6</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Interests that Google Stores about Me

We use interests from your activity on websites to tailor ads to you. How it works. Please note the listed categories do not include some of the ways ads may be tailored to you, including remarketing lists based on visits to advertiser websites.

- Greece
- Health Insurance
- Inheritance & Estate Planning
- Insurance
- Investing
- Parenting
- Performing Arts
- Politics
- Primary & Secondary Schooling (K-12)
- Search Engines
- Soccer
- Social Networks
- Travel
- Water Sports
- Weather

https://www.google.com/ads/preferences/
Example: Data Reduction

Only a subset of the location data sent by Android phone is stored.

https://maps.google.com/locationhistory/
3. Web Usage Mining Tasks

1. Website Personalization
   - Personalized content and navigation elements
   - Optimization of website structure
   - Techniques: Classification, Sequential Pattern Mining

2. Marketing
   - Discovery of associated products for cross-selling
     - Association rules, Sequential Pattern Mining
     - Placement of associated products on the same page
   - Discovery of associated products in different price categories for up-selling
     - Association rules, Sequential Pattern Mining
   - Identification of Customer Groups
     - Clustering, Classification
   - Formulation of personalized recommendations
     - Suggestions of similar items (e.g. pages or products)
     - Suggestions of items based on the preferences of similar users
Tasks and Techniques

- Prediction of the next event
- Discovery of associated events or application objects
- Recommendation of products and content
- Discovery of visitor groups with common properties and interests
- Discovery of visitor groups with common behaviour
- Characterization of visitors into predefined classes
- Card fraud detection

Techniques:
- Sequence mining
- Markov chains
- Association rules
- Recommendation Algorithms
- Clustering
- Session Clustering
- Classification

Prediction of the next event
- Discovery of associated events or application objects
- Recommendation of products and content
- Discovery of visitor groups with common properties and interests
- Discovery of visitor groups with common behaviour
- Characterization of visitors into predefined classes
- Card fraud detection
References for the Basic Techniques

- **Clustering**
  - Slide set: Lecture IE500: Data Mining I – Chapter 2: Clustering
  - Book: Tan, Steinbach: Introduction to Data Mining, Chapters 8, 9

- **Classification**
  - Slide set: Lecture IE500: Data Mining I – Chapter 3: Classification
  - Book: Tan, Steinbach: Introduction to Data Mining, Chapters 4, 5

- **Association Analysis and Sequential Patterns**
  - Slide set: Lecture IE500: Data Mining I – Chapter 4: Association Analysis
  - Slide set: Lecture IE672: Data Mining II – Chapter 3: Time Series Analysis
  - Book: Tan, Steinbach: Introduction to Data Mining, Chapters 6, 7

- **Tools:**
  - Single machine: Rapidminer
  - Cluster with multiple machines: Apache Mahout
4. Recommender Systems

Recommender Systems (RS) help to match users with items

- ease information overload
- sales assistance (guidance, advisory, persuasion, ...)

Recommender Systems can be seen as a function

Given:
- User model (e.g. ratings, preferences, demographics, situational context)
- Items (with or without description of item characteristics)

Find:
- Relevance score. Used for determining Top-K items.

Concrete system design depends

- on availability of exploitable data
- implicit and explicit user feedback
- domain characteristics
Application Domains of Recommender Systems

- Which movie should I watch?
- Which digital camera should I buy?
- Which news article will I find interesting?
- Toward which degree should I study? –
- Which is the best investment for my retirement money? –
Paradigms of Recommender Systems

- **Demographic Recommendation**
  - Offer Backstreet Boys albums only to girls under 16
  - Offer cameras with American electricity plug to people from US.

- **Contextual Recommendation (Location / Time of Day/Year)**
  - Send coupon to mobile user who passes by a shop (Foursquare)
  - Show holiday related advertisements based on user location
Paradigms of Recommender Systems

Collaborative: "Tell me what's popular among my peers"

User–Item Rating Matrix

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Paradigms of Recommender Systems

Content-based: "Show me more of the same what I've liked"
Paradigms of Recommender Systems

Hybrid: Combinations of various inputs and/or composition of different mechanisms
1. User’s Perspective
   - Recommend me items that I like and did not know about
   - Serendipity: Accident of finding something good while not specifically searching for it

2. Merchant’s Perspective
   - Increase the sale of high-revenue items
   - Thus real-world recommender systems are not as neutral as the following slides suggest
4.1 Collaborative Filtering

- The most prominent approach to generate recommendations
  - used by large e-commerce sites
  - applicable in many domains (book, movies, DVDs, ..)

- Approach
  - use the "wisdom of the crowd" to recommend items

- Basic Assumptions
  1. Users give ratings to catalog items (implicitly or explicitly)
  2. Customers who had similar tastes in the past, will have similar tastes in the future

- Input: Matrix of given user–item ratings

- Output types
  1. (Numerical) prediction indicating to what degree the current user will like or dislike a certain item
  2. Top-K list of recommended items
Given an "active user" (Alice) and an item $i$ not yet rated by Alice

1. find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past and who have rated item $i$
2. use their ratings of item $i$ to predict, if Alice will like item $i$
3. do this for all items Alice has not seen and recommend the best-rated.

Example: User–Item Rating Matrix

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: >99% of real-world values are NULL
User-Based Nearest-Neighbor Collaborative Filtering

Some questions we need to answer

1. How do we measure user similarity?
2. How many neighbors should we consider?
3. How do we generate a prediction from the neighbors' ratings?

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Measuring User Similarity

A popular similarity measure in user-based CF is the Pearson Correlation Coefficient:

\[ sim(a, b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}} \]

- Takes different usage of rating scale into account by comparing individual ratings to the user’s average rating.
- Value range [-1,1]
  - 1 : positive correlation
  - 0 : variables independent
  - -1 : negative correlation
Example: Pearson Correlation

A popular similarity measure in user-based CF is the Pearson Correlation Coefficient

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

sim = 0.85
sim = 0.70
sim = 0.00
sim = -0.79
Pearson Correlation

- Takes differences in rating behavior into account.
- Some people always give higher ratings than others.

Empirical studies show that Pearson Correlation often works better than alternative measures such as cosine similarity.
Making Predictions

1. A simple prediction function:

\[ \text{pred}(a, p) = \frac{\sum_{b \in N} \text{sim}(a, b) \times r_{b, p}}{\sum_{b \in N} \text{sim}(a, b)} \]

- Uses the similarity with \( a \) as a weight to combine ratings

2. A prediction function that takes rating behavior into account:

\[ \text{pred}(a, p) = \bar{r}_a + \frac{\sum_{b \in N} \text{sim}(a, b) \times (r_{b, p} - \bar{r}_b)}{\sum_{b \in N} \text{sim}(a, b)} \]

- Calculates whether the neighbors' ratings for the unseen item \( i \) are higher or lower than their average
- Uses the similarity with \( a \) as a weight to combine rating differences
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction
Improving the Metrics / Prediction Function

- Neighborhood Selection
  - Use fixed number of neighbors or similarity threshold

- Case Amplification
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
  - Implementation: $\text{sim}(a, b)^2$

- Rating Variance
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - Possible solution: Give more weight to items that have a higher variance
Memory-based and Model-based Approaches

- **User-based CF is said to be "memory-based"**
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios as large e-commerce sites have tens of millions of customers and 10,000s of items

- **Model-based approaches**
  - employ offline model-learning
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - A large variety of techniques is used
    1. Item-based Collaborative Filtering
    2. Association Rules
    3. Probabilistic Methods
    4. Matrix Factorization Techniques

See: IE 673: Data Mining and Matrices
Item-based Collaborative Filtering

**Basic idea:**
- Use the similarity between items (and not users) to make predictions

**Approach:**
1. Look for items that have been rated similarly as Item5
2. Take Alice's ratings for these items to predict the rating for Item5

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Calculating Item-to-Item Similarity

- **Cosine Similarity**
  - Often produces better results as Pearson for calculating the item-to-item similarity.
  
  \[
  \text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \times |\vec{b}|}
  \]

  \[
  \vec{a} \cdot \vec{b} = \sum_{i=1}^{N} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n
  \]

  \[
  |\vec{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}
  \]

- **Adjusted Cosine Similarity**
  - Adjusts ratings by taking the average rating behavior of a user into account.
  - \( U \): set of users who have rated both items \( a \) and \( b \)

  \[
  \text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}
  \]
Making Predictions

A common prediction function for item-based CF:

\[
\text{pred}(u, p) = \frac{\sum_{i \in \text{ratedItem}(u)} \text{sim}(i, p) \cdot r_{u,i}}{\sum_{i \in \text{ratedItem}(u)} \text{sim}(i, p)}
\]

\text{ratedItem}(u) : Set of items rated by Alice

\text{r}_{ui} : Alice’s rating for items i

\text{sim}(i, p) : Similarity of item i with target item p
Pre-Processing for Item-Based Filtering

- Item-based filtering does not solve the scalability problem itself, but as there are usually less items than users, we can pre-calculate the item similarities and store them in memory.

- Neighborhood size is typically also limited to a specific size
  - An analysis of the MovieLens dataset indicates a neighborhood size of 20 to 50 items is reasonable (Herlocker et al. 2002)
  - Not all neighbors are taken into account for the prediction, as Alice most likely only rated a small subset of the neighbors

- Memory requirements
  - Up to $N^2$ pair-wise similarities to be memorized ($N =$ number of items) in theory
  - In practice, the memory requirements are significantly lower as
    - many items have no co-ratings (heavy metal and samba CDs)
    - neighborhood size often limited to max items above minimum similarity threshold
Recap: Association Rule Mining

- Commonly used for shopping basket analysis
  - aims at detection of rules such as "If a customer purchases beer then he also buys diapers in 70% of the cases"

- Association rule mining algorithms
  - detect rules of the form $X \rightarrow Y$ (e.g., beer $\rightarrow$ diapers) from a set of sales transactions $D = \{t_1, t_2, \ldots, t_n\}$
  - Two step rule mining process
    1. determine frequent item sets
    2. derive rules from the frequent item sets

- Measures of rule quality
  - used e.g. as a threshold to cut off unimportant rules
  - Support count = $\sigma(X) = |\{x | x \subseteq t_i, t_i \in D\}|$
  - Support = $\frac{\sigma(X \cup Y)}{|D|}$
  - Confidence = $\frac{\sigma(X \cup Y)}{\sigma(X)}$

See: IE500 Data Mining: Chapter 4
Un-Personalized Recommendation

Items co-occurring with book in frequent item sets

[Image of an Amazon page showing recommendations related to a book on Linked Data]
Personalized Recommendation using Association Rules

- **Simplest approach**
  - transform 5-point ratings into binary ratings (1 = above user average)

- **Mine rules such as**
  - Item1 → Item5
    - support (2/4), confidence (2/2) (without Alice)

- **Make recommendations for Alice (basic method)**
  1. determine "relevant" rules based on Alice's transactions/ratings (the above rule will be relevant as Alice bought/rated Item1)
  2. determine items not already bought/rated by Alice
  3. sort the items based on the rules' confidence values

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>User2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>User3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>User4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Probabilistic Methods

Basic idea:
- given the user/item rating matrix
- determine the probability that Alice will give item $i$ a specific rating

Calculation of rating probabilities based on Bayes Theorem
- Given Alice's previous ratings, how probable is it that she rates Item5 with the rating value 1?
- Corresponds to conditional probability $P(\text{Item5}=1 \mid X)$, where $X = \text{Alice's previous ratings} = (\text{Item1}=1, \text{Item2}=3, \text{Item3}=\ldots)$
- Can be estimated using Bayes' theorem and independence assumption $Y = \text{Item5}=1$

$$P(Y \mid X) = \frac{P(X \mid Y) \times P(Y)}{P(X)}$$

See: IE500 Data Mining: Chapter 3
Estimation of the Probabilities

\[ X = (\text{Item1}=1, \text{Item2}=3, \text{Item3}= \ldots ) \]

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>User2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>User3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
P(X|\text{Item5} = 1) = P(\text{Item1} = 1|\text{Item5} = 1) \times P(\text{Item2} = 3|\text{Item5} = 1) \times P(\text{Item3} = 3|\text{Item5} = 1) \times P(\text{Item4} = 2|\text{Item5} = 1) \approx 0.125
\]

- Probabilities can be estimated (counted) from the user-rating-matrix
- Empirical analysis shows that probabilistic methods lead to relatively good results (movie domain)
- Small memory-footprint of leaned model
More on Ratings: Explicit Ratings

- Explicit ratings are probably the most precise ratings.

- Commonly used response scales:
  - 1 to 5 Likert scales
  - Like (sometimes also Dislike)

- Main problems
  - Users often not willing to rate items
    - number of ratings likely to be too small → poor recommendation quality
  - How to stimulate users to rate more items?
    - Example: Amazon Betterizer

- Alternative
  - Use implicit ratings (in addition to explicit ones)
More on Ratings: Implicit Ratings

- Events potentially interpretable as positive ratings
  - items bought
  - clicks, page views
  - time spent on some page
  - demo downloads …

- Advantage
  - implicit ratings can be collected constantly by the web site or application in which the recommender system is embedded
  - collection of ratings does not require additional effort from the user

- Problem
  - one cannot be sure whether the user behavior is correctly interpreted
  - for example, a user might not like all the books he or she has bought; the user also might have bought a book for someone else
Collaborative Filtering Discussion

Pros:
- works well in some domains: books, movies. Likely not: life insurances
- requires no explicit item descriptions or demographic user profiles

Cons:
- requires user community to give enough ratings (many real-world systems thus employ implicit ratings)
- no exploitation of other sources of recommendation knowledge (demographic data, item descriptions)

Cold Start Problem
- How to recommend new items?
- What to recommend to new users?

Approaches for dealing with the Cold Start Problem
- Ask/force users to rate a set of items
- Use another method or combination of methods (e.g., content-based, demographic or simply non-personalized) until enough ratings are collected
4.2 Content-based Recommendation

- While collaborative filtering methods do not use any information about the items, it might be reasonable to exploit such information.
  - e.g., recommend fantasy novels to people who liked fantasy novels in the past

- What do we need:
  - information about the available items (content)
  - some sort of user profile describing what the user likes (user preferences)

- The tasks:
  1. learn user preferences from what she has bought/seen before
  2. recommend items that are "similar" to the user preferences
### Content and User Profile Representation

#### Content Representation

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Lace Reader</td>
<td>Fiction, Mystery</td>
<td>Brunonia Barry</td>
<td>Hardcover</td>
<td>49.90</td>
<td>American contemporary fiction, detective, historical</td>
</tr>
<tr>
<td>Into the Fire</td>
<td>Romance, Suspense</td>
<td>Suzanne Brockmann</td>
<td>Hardcover</td>
<td>45.90</td>
<td>American fiction, murder, neo-nazism</td>
</tr>
</tbody>
</table>

#### User Profile

<table>
<thead>
<tr>
<th>Title</th>
<th>Genres</th>
<th>Authors</th>
<th>Types</th>
<th>Avg. Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>Fiction, Mystery</td>
<td>Brunonia, Barry, Ken Follett</td>
<td>Paperback</td>
<td>25.65</td>
<td>Detective, murder, New York</td>
</tr>
</tbody>
</table>

#### Simple recommendation approach

- Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)

\[
2 \times \frac{|\text{keywords}(b_i) \cap \text{keywords}(u)|}{|\text{keywords}(b_i)| + |\text{keywords}(u)|}
\]

#### More sophisticated approach

- use attribute specific similarity measures and weights

See: IE500 Data Mining: Chapter 2 (Proximity)
**Recommending Text Documents**

- Content-based recommendation techniques are often applied to recommend text documents, like news articles or blog posts.
- Documents and user profiles are represented as *term-vectors*:

<table>
<thead>
<tr>
<th>Document Corpus</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>157</td>
<td>73</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td>157</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>232</td>
<td>227</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>10</td>
<td>123</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>17</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>0</td>
<td>43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Profile</th>
<th>Liked Doc 1</th>
<th>Liked Doc 2</th>
<th>Liked Doc 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>233</td>
<td>99</td>
<td>34</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>22</td>
<td>23</td>
<td>90</td>
</tr>
</tbody>
</table>
Similarity of Text Documents

- **Challenges**
  - terms vectors are very sparse
  - not every word has the same importance
  - long documents have higher chance to overlap with user profile

- **Methods for handling these challenges**
  - Similarity metric: Cosine similarity
  - Preprocessing: remove stop words
  - Vector Creation:
    Term-Frequency - Inverse Document Frequency \((TF - IDF)\)

See: IE500
Data Mining: Chapter 6
Recommending Documents

- Given a set of documents $D$ already rated by the user
  - either explicitly via user interface
  - or implicitly by monitoring user behavior

1. Find the $n$ nearest neighbors of an not-yet-seen item $i$ in $D$
   - measure similarity of item $i$ with neighbors using cosine similarity

2. Use ratings from Alice for neighbors to predict a rating for item $i$
   - Find 5 most similar items to $i$
   - 4 of these items were liked by Alice $\Rightarrow$ item $i$ will also be liked by Alice

Variations:
- Varying neighborhood size $k$
- upper similarity threshold to prevent system from recommending too similar texts (variations of texts the user has already seen)

Good to model short-term interests / follow-up stories

Often used in combination with method to model long-term preferences
- E.g. ‘Semantic enrichment’ by assigning interests to each page/product.
Content-based Filtering Discussion

**Pros:**
- In contrast to collaborative approaches, content-based techniques do not require user community in order to work
- No problems with recommending new items

**Cons:**
- Require to learn a suitable model of user's preferences based on explicit or implicit feedback
  - deriving implicit feedback from user behavior can be problematic
  - ramp-up phase required (users needs to view/rate some items)
  - Web 2.0: Use other sources to learn the user preferences might be an option (e.g. share your Facebook profile with e-shop)
- Overspecialization
  - Algorithms tend to propose "more of the same"
  - Recommendations might be boring as items are too similar
4.3 Hybrid Recommender Systems

Hybrid: Combinations of various inputs and/or composition of different mechanism in order to overcome problems of single methods.

Demographic: “Offer American plugs to people from the US“

Collaborative: "Tell me what's popular among my peers"

Content-based: "Show me more of the same what I've liked"
Parallelized Hybridization Design

- Output of several existing recommenders is combined
- Least invasive design
- Requires some weighting or voting scheme
  - weights can be learned using existing ratings as supervision
  - dynamic weighting: Adjust weights or switch between different recommenders as more information about users and items becomes available
    - e.g. if too few ratings available the use content-based recommendation, otherwise use collaborative filtering
Parallelized Hybridization Design: Weighted

- Compute weighted sum:

\[ \text{rec}_{\text{weighted}}(u,i) = \sum_{k=1}^{n} \beta_k \times \text{rec}_k(u,i) \]

<table>
<thead>
<tr>
<th>Recommender 1</th>
<th>Item1</th>
<th>0.5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item3</td>
<td>0.3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Item4</td>
<td>0.1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Item5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommender 2</th>
<th>Item1</th>
<th>0.8</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item2</td>
<td>0.9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Item3</td>
<td>0.4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Item4</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommender weighted(0.5:0.5)</th>
<th>Item1</th>
<th>0.65</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item2</td>
<td>0.45</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Item3</td>
<td>0.35</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Item4</td>
<td>0.05</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Item5</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adjustment of Weights

- Use existing ratings to learn individual weights for each user
- Compare prediction of recommenders with actual ratings by user
- For each user, adapt weights to minimize Mean Absolute Error (MAE)

\[
MAE = \frac{\sum_{i \in R} \sum_{k=1}^{n} \beta_k \times |rec_k(u,i) - r_i|}{|R|}
\]

MAE improves as rec2 is weighted more strongly

<table>
<thead>
<tr>
<th>Absolute errors and MAE</th>
<th>rec1</th>
<th>rec2</th>
<th>error</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight1: 0.1, Weight2: 0.9</td>
<td>Item1: 0.5</td>
<td>0.8</td>
<td>0.23</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Item4: 0.1</td>
<td>0.0</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>Weight1: 0.3, Weight2: 0.7</td>
<td>Item1: 0.5</td>
<td>0.8</td>
<td>0.29</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>Item4: 0.1</td>
<td>0.0</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>Weight1: 0.5, Weight2: 0.5</td>
<td>Item1: 0.5</td>
<td>0.8</td>
<td>0.35</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>Item4: 0.1</td>
<td>0.0</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>Weight1: 0.7, Weight2: 0.3</td>
<td>Item1: 0.5</td>
<td>0.8</td>
<td>0.41</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>Item4: 0.1</td>
<td>0.0</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>Weight1: 0.9, Weight2: 0.1</td>
<td>Item1: 0.5</td>
<td>0.8</td>
<td>0.47</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>Item4: 0.1</td>
<td>0.0</td>
<td>0.91</td>
<td></td>
</tr>
</tbody>
</table>
Monolithic Hybridization Design

- Features/knowledge sources of different paradigms are combined in a single recommendation component. E.g.:
  - Ratings and user demographics
  - Ratings and content features: user likes many movies that are comedies

Example: Content-boosted Collaborative Filtering

- based on content features additional ratings are created
- e.g. Alice likes Items 1 and 3 (unary ratings)
  - Item7 is similar to 1 and 3 by a degree of 0.75
  - Thus Alice likes Item7 by 0.75
- rating matrix becomes less sparse
- see [Prem Melville, et al. 2002]
4.4 Evaluating Recommender Systems

Question: Is a Recommender System efficient with respect to a specific criteria like accuracy, serendipity, online conversion, response time, ramp-up efforts?

So we need to determine the criteria that matter to us

Popular Measures for Accuracy

- If items are rated on a Likert scale (1 to 5)
  - MAE (Mean Absolute Error), RMSE (Root Mean Squared Error)
- If items are classified as good or bad
  - Precision / Recall / F1-Score
- If items are presented as ranked Top-K list
  - Lift Index, Normalized Discounted Cumulative Gain

Methodologies for measuring Accuracy

- Split-Validation, Cross-Validation
Evaluation Methodology

- **Setting to ensure internal validity:**
  - One randomly selected share of known ratings (training set) used as input to train the algorithm and build the model.
  - Remaining share of withheld ratings (test set) used as ground truth to evaluate quality.
  - To ensure the reliability of measurements the random split, model building and evaluation steps are repeated several times.

- **Split-Validation**
  - Split-Validation: e.g. 2/3 training, 1/3 validation.

- **N-Fold Cross Validation**
  - N disjunct fractions of known ratings with equal size (1/N) are determined. Setting N to 5 or 10 is popular.
  - N repetitions of the model building and evaluation steps, where each fraction is used exactly once as a testing set while the other fractions are used for training.

<table>
<thead>
<tr>
<th>Item</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item1</td>
<td>5</td>
</tr>
<tr>
<td>Item2</td>
<td>1</td>
</tr>
<tr>
<td>Item3</td>
<td>3</td>
</tr>
<tr>
<td>Item4</td>
<td>1</td>
</tr>
<tr>
<td>Item5</td>
<td>4</td>
</tr>
<tr>
<td>Item6</td>
<td>2</td>
</tr>
</tbody>
</table>

Alice

Training Set

Test Set
Evaluation of Likert-Scaled Predictions

- **Mean Absolute Error (MAE)** computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |p_i - r_i|$$

- **Root Mean Square Error (RMSE)** is similar to MAE, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (p_i - r_i)^2}$$

- **Critique**
  - Not meaningful as inclusion into Top-K list is more important to the user than overall accuracy of predictions.
  - Rather evaluate inclusion into Top-K list as classification problem (see next slide).
Evaluation of Good/Bad Classifications

- **Precision**: Measure of exactness.
  - determines the fraction of relevant items retrieved out of all items retrieved
  - fraction of recommended movies that are actually good

- **Recall**: Measure of completeness.
  - determines the fraction of relevant items retrieved out of all relevant items
  - E.g. the fraction of all good movies recommended

- **F1-Measure**
  - combines Precision and Recall into a single value for comparison purposes.
  - May be used to gain a more balanced view of performance

**Confusion Matrix**

<table>
<thead>
<tr>
<th></th>
<th>Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actually Good</td>
</tr>
<tr>
<td><strong>Prediction</strong></td>
<td><strong>Good</strong></td>
</tr>
<tr>
<td>Good</td>
<td><strong>True Positive</strong></td>
</tr>
<tr>
<td></td>
<td>(tp)</td>
</tr>
<tr>
<td>Bad</td>
<td><strong>False Negative</strong></td>
</tr>
<tr>
<td></td>
<td>(fn)</td>
</tr>
</tbody>
</table>

\[
Precision = \frac{tp}{tp + fp} = \frac{|good \ movies \ recommended|}{|all \ recommendations|}
\]

\[
Recall = \frac{tp}{tp + fn} = \frac{|good \ movies \ recommended|}{|all \ good \ movies|}
\]

\[
F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}
\]
Rank position also matters!

Rank metrics extend recall and precision to take the positions of correct items in a ranked list into account

- Relevant items are more useful when they appear earlier in the recommendation list
- Particularly important in recommender systems as lower ranked items may be overlooked by users

#### Evaluation of ranked Top-K List

For a specific user:

<table>
<thead>
<tr>
<th>Actually good</th>
<th>Recommended (predicted as good)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item 237</td>
<td>Item 345</td>
</tr>
<tr>
<td>Item 899</td>
<td>Item 237</td>
</tr>
<tr>
<td></td>
<td>Item 187</td>
</tr>
</tbody>
</table>
Lift Index

- Ranked list is divided into 10 equal deciles $S$
  - $h$ is the set of correct hits
  - $S_i$ number of correct hits in decile

\[ \sum_{i=1}^{10} S_i = |h| \]

- Lift Index
  - Uses a linear reduction factor

\[
\text{liftindex} = \begin{cases} 
  \frac{1 \times S_1 + 0.9 \times S_2 + \ldots + 0.1 \times S_{10}}{\sum_{i=1}^{10} S_i} : \text{if } |h| > 0 \\
  0 : \text{else}
\end{cases}
\]

- Example

\[
\text{liftindex} = \frac{0.9 \times 1 + 0.8 \times 1 + 0.7 \times 1}{3} = 0.8
\]

<table>
<thead>
<tr>
<th>Rank</th>
<th>Hit?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Normalized Discounted Cumulative Gain

**Discounted cumulative gain (DCG)**

- Uses a logarithmic reduction factor

  \[ DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i} \]

  where:
  - pos denotes the position up to which relevance is accumulated
  - rel\(_i\) returns the relevance (0/1) of recommendation at position \(i\)

**Idealized discounted cumulative gain (IDCG)**

- Assumption that all good items are ordered by decreasing relevance

  \[ IDCG_{pos} = rel_1 + \sum_{i=2}^{|h|-1} \frac{rel_i}{\log_2 i} \]

**Normalized discounted cumulative gain (nDCG)**

- Normalized to the interval [0..1]

  \[ nDCG_{pos} = \frac{DCG_{pos}}{IDCG_{pos}} \]

**Example**

\[
DCG_5 = \frac{1}{\log_2 2} + \frac{1}{\log_2 3} + \frac{1}{\log_2 4} = 2.13 \\
IDCG_5 = 1 + \frac{1}{\log_2 2} + \frac{1}{\log_2 3} = 2.63 \\
nDCG_5 \frac{DCG_5}{IDCG_5} \approx 0.81
\]
4.5 Attacks on Recommender Systems

- As there is (monetary) value in being on recommendation lists
  - individuals/companies may be interested to push or nuke some items by manipulating the recommender system

- Basic Attack Strategies
  - automatically create numerous fake accounts / profiles
  - issue high or low ratings for target item
  - rate additional filler items in order to
    - make fake profile appear in neighborhood of many real-world users and
    - camouflage fake profiles
  - for implicit ratings: Use crawler that automatically navigates the site

- Countermeasures
  1. make it difficult to generate fake profiles (e.g. using Captchas)
  2. use machine-learning methods to discriminate real from fake profiles

- Details on attacks and countermeasures
  - Jannach et al.: Recommender Systems. Chapter 9
Public Rating Datasets

- **MovieLens**
  - movie ratings collected via MovieLens website
  - 1M Dataset: 6,000 users, 3,900 movies, 1 million ratings
  - 10M Dataset: 71,000 users, 10,600 movies, 10 million ratings

- **Netflix**
  - provided by commercial movie rental website for Netflix competition ($1,000,000 for 10% better RMSE)
  - 480,000 users rated 18,000 movies, 100M ratings

- **Yahoo Music**
  - 600,000 songs, 1 million users, 300M ratings
  - provided for KDD Cup 2011

- **Web 2.0 Platforms offer plenty of additional rating data**
  - e.g. LastFM, delicious
Exercise

- You experiment with different recommendation algorithms using Mahout.
- Please bring your laptop!

- open-source machine learning library which can be deployed on a single machine as well as on Hadoop and Spark clusters
- includes various recommendation algorithms
- Download: http://mahout.apache.org/