Web Mining

Web Usage Mining and Recommender Systems – Part 1 –

Prof. Dr. Christian Bizer

FSS 2023
Web Usage Mining

**Definition**

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

**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)

**Associated Data**

1. page attributes, page content, site structure
2. additional domain knowledge and demographic data
3. user profiles or user ratings
Web Usage Data: The Oil of the New Economy

2021 This Is What Happens In An Internet Minute

Google Analytics

Get Traffic Analysis

Provide Access to

Web Server Logs

Created By: @LoriLewis @OfficiallyChadd

Economic and Social Impact of Usage Data Collection

- Who owns the usage data?
  - the user? private companies? government?

- Who is allowed to use it for what?
  - Companies for targeting users?
  - Government for fighting COVID?
  - Government for law enforcement?

- Privacy law, and yes boxes

- Alternative: SOLID
  - decentral data collection and decentral rights tracking
  - difficult to deploy
  - https://solidproject.org/

Google COVID Lockdown Movement Tracking

Social Scoring of „trustworthiness“
The Web Usage Mining Process
Chapter Outline

1. Usage Data Collection
2. Usage Data Preparation
   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. Model-based Recommendation
   4. Hybrid Recommendation
   5. Evaluating Recommender Systems
   6. Attacks on Recommender Systems
1. Usage Data Collection

**Server-Side Data Collection**
- Traditional web server logs
  - Content: IP, timestamp, page URL, 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
  - often not restricted to single server
- via providing the application
- additional collectable data:
  - mouse movements
  - keyboard strokes
  - size of browser window
Recording Users Entering and Leaving the Site

Web server logs may extend beyond visits to the site and show

- where a visitor was before (via HTTP **Referer**)
  
  203.30.5.145 - - [01/Jun/2021:03:09:21 -0600] "GET /Calls/OWOM.html HTTP/1.0"
  200 3942 "http://www.lycos.com/cgi-bin/pursuit?query=advertising+psychology-
  &maxhits=20&cat=dir" "Mozilla[en] (Win10; I)"

- and where she went next (via **URL Rewriting**):
  
  often used be search engines to get user feedback about search results
## 2. Data Preparation

### Content of a typical Apache web server log:

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Date</th>
<th>Method</th>
<th>File</th>
<th>Protocol</th>
<th>Status Code</th>
<th>Bytes</th>
<th>Referer</th>
<th>User Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>203.30.5.145</td>
<td>[01/Jul/2021]</td>
<td>GET</td>
<td>/Calls/OWOM.html</td>
<td>HTTP/1.0</td>
<td>200</td>
<td>3942</td>
<td><a href="http://www.lycos.com/cgi-bin/pursuit?query=advertising+psychology-">http://www.lycos.com/cgi-bin/pursuit?query=advertising+psychology-</a> &amp;maxhits=20&amp;cat=dir</td>
<td>Mozilla/4.5 [en] (Win98; I)</td>
</tr>
<tr>
<td>203.30.5.145</td>
<td>[01/Jul/2021]</td>
<td>GET</td>
<td>/Calls/Images/earthani.gif</td>
<td>HTTP/1.0</td>
<td>200</td>
<td>10689</td>
<td><a href="http://www.acr-news.org/Calls/OWOM.html">http://www.acr-news.org/Calls/OWOM.html</a></td>
<td>Mozilla/4.5 [en] (Win98; I)</td>
</tr>
<tr>
<td>203.252.234.33</td>
<td>[01/Jul/2021]</td>
<td>GET</td>
<td>/</td>
<td>HTTP/1.0</td>
<td>200</td>
<td>4980</td>
<td></td>
<td>Mozilla/4.06 [en] (Win95; I)</td>
</tr>
<tr>
<td>203.252.234.33</td>
<td>[01/Jul/2021]</td>
<td>GET</td>
<td>/Images/line.gif</td>
<td>HTTP/1.0</td>
<td>200</td>
<td>190</td>
<td><a href="http://www.acr-news.org/">http://www.acr-news.org/</a></td>
<td>Mozilla/4.06 [en] (Win95; I)</td>
</tr>
<tr>
<td>203.252.234.33</td>
<td>[01/Jul/2021]</td>
<td>GET</td>
<td>/Images/red.gif</td>
<td>HTTP/1.0</td>
<td>200</td>
<td>104</td>
<td><a href="http://www.acr-news.org/">http://www.acr-news.org/</a></td>
<td>Mozilla/4.06 [en] (Win95; I)</td>
</tr>
<tr>
<td>203.252.234.33</td>
<td>[01/Jul/2021]</td>
<td>GET</td>
<td>/Images/earthani.gif</td>
<td>HTTP/1.0</td>
<td>200</td>
<td>10689</td>
<td><a href="http://www.acr-news.org/">http://www.acr-news.org/</a></td>
<td>Mozilla/4.06 [en] (Win95; I)</td>
</tr>
</tbody>
</table>
Data Preparation

1. **Data Cleansing**
   - remove irrelevant log entries and fields from server logs
     - usually: remove all log entries related to images or scripts
     - ignoring certain page-views / items
   - remove log entries due to crawler navigation (>50% of all requests)

2. **Data Integration**
   - synchronize data from multiple server logs (due to server farms)
   - 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 / semantic enrichment

4. **Data Reduction**
   - sampling
Robot Detection

1. Identification via HTTP User-Agent Header
   - using list of known robots, e.g. from http://useragentstring.com/

2. Classification using Behavioural Features
   - Accesses robots.txt file
   - time on page
   - navigation patterns
   - no download of images or scripts

Example of Web Crawler Traffic

Tan, Kumar: Discovery of Web Robot Sessions based on their Navigational Patterns. Data Mining and Knowledge Discovery 6(1), 2002.
## 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: apps, browsers, page tags (use javascript)
Mechanisms for Session Identification

**Time oriented heuristics**

- $15/\text{Dec}/2000:17:01:41$

**Navigation oriented heuristic**

- [http://iwa.wiwi.hu-berlin.de/X.html](http://iwa.wiwi.hu-berlin.de/X.html)

**h1**: Total session duration must not exceed a maximum

- 30 minutes

**h2**: Page stay times must not exceed a maximum

- 10 minutes

**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

**threshold**

- in the experiments reported here

Source: Spiliopoulou et al., 2003
Data Aggregation

- aggregate log data in order to generate features that are suitable for the task at hand (identify robots, cluster users, …)

- 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>
## Data Aggregation

- **Example of a User Pageview Matrix**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>user0</td>
<td>15</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>185</td>
</tr>
<tr>
<td>user1</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>user2</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>236</td>
<td>0</td>
</tr>
<tr>
<td>user3</td>
<td>9</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>134</td>
</tr>
<tr>
<td>user4</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>user5</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>157</td>
<td>69</td>
<td>0</td>
</tr>
<tr>
<td>user6</td>
<td>24</td>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>354</td>
</tr>
<tr>
<td>user7</td>
<td>0</td>
<td>0</td>
<td>78</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>user8</td>
<td>7</td>
<td>0</td>
<td>45</td>
<td>20</td>
<td>127</td>
<td>0</td>
</tr>
<tr>
<td>user9</td>
<td>0</td>
<td>38</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>

- **Useful for discovering user groups (cluster analysis)**
Semantic Enrichment

■ Basic Idea

Associate each requested page with one or more topics/concepts to better understand user behavior.

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

■ Aggregation Levels:
  ■ Page level: 1 request ➔ 1 concept or n concepts for example: insurances, travel, …
  ■ Session level: set / sequence of pages ➔ 1 concept or n concepts for example: user compares insurance offers

■ Concepts can be part of a concept hierarchy or ontology:
  ■ Useful for building/maintaining user profiles
Example: Semantic Enrichment

- **Input: User Pageview Matrix**

- **Input: Page Topic Matrix**

- **Result: User Topic Matrix**
Interests that Google Stores about Me

https://adssettings.google.com/
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
   - Techniques: Classification, Re-Ranking, 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 for Targeted Marketing
     - Clustering, Classification
   - Personalized recommendations
     - Suggestions of similar items (e.g. pages or products)
     - Suggestions of items based on the preferences of similar users
Summary: Usage Mining 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

- Sequential patterns
- Association rules
- Markov chains
- Recommender Systems
- User Clustering
- Session Clustering
- Classification
2. 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/rating score. Used for determining the top-k items

- Concrete system design depends on
  - the availability of exploitable data
  - domain characteristics
Application Domains of Recommender Systems

- Which music will I like?
- Which movie should I watch?
- Which book should I buy?
- Which news fit to my political position? (Filter bubbles)
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
Paradigms of Recommender Systems

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

- **Demographic Recommendation**
  - offer cameras with American electricity plug to people from US
  - offer Backstreet Boys albums to people under the age of 16

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

Hybrid: Combinations of various inputs and/or composition of different mechanisms
2.1 Collaborative Filtering

- **A standard approach to generate recommendations**
  - used by large e-commerce sites
  - applicable in many domains (book, movies, DVDs, ..)

- **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 (i.e., a rating itself)
  2. Ranking: Top-k list of recommended items
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 small
    → poor recommendation quality
  - How to stimulate users to rate more items?
    - Example: Amazon Betterizer

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

- Events potentially interpretable as positive ratings
  - items bought
  - clicks, page views
  - time spent on some page
  - time a movie was watched …

- 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

- Most deployed recommender systems rely on implicit ratings
User-Based Nearest-Neighbor Collaborative Filtering

- 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 top-rated $k$ items

- 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>

See: Data Mining I: KNN
Questions we need to answer

1. How do we measure user similarity?
   - given that real-world user/item matrices are very sparse (>90% missing values)

2. How many neighbors should we consider?
   - hyperparameter k in KNN

3. How do we generate a prediction from the neighbors' ratings?
   - given that different people use the rating scale differently

<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></td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td></td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>User3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User4</td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
A popular similarity measure in user-based CF is the **Pearson Correlation Coefficient**

- **a, b**: users
- **r_{a,p}**: rating of user a for item p
- **P**: set of items, **rated by both** a and b

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

- Takes different usage of rating scale into account by comparing individual ratings to the user’s average rating
- Note: For Pearson you need **paired data**, that is, we take only the ratings for the set of items, rated by both users (also to compute the average ratings)
Example: Measuring User Similarity

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

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

- \( a, b \) : users
- \( r_{a,p} \) : rating of user \( a \) for item \( p \)
- \( P \) : set of items, rated by both \( a \) and \( b \)

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>Item5</th>
<th>sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
<td>0.85</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0.70</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>-0.79</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Making Predictions

1. A simple prediction function:

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

- uses the similarity with \( a \) as a weight to combine ratings
- \( N \) is the number of similar users that should be considered (hyperparameter)

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

\[
pred(a, p) = \frac{\sum_{b \in N} sim(a, b) \times (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a, b)} + \overline{r_a}
\]

- 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
Example: Making Predictions

- To make a prediction for Item5, we first decide which of the neighbours’ ratings we take into account and apply the second formula from the previous slide.

- In our example, an obvious choice would be to take User1 and User2 as peer users to predict Alice’s rating.

- Hence the prediction for Alice’s rating for Item5 based on the ratings of nearest neighbours User1 and User2 will be

$$\text{pred}(\text{Alice, Item5}) = 4 + \left( \frac{0.85 \cdot (3 - 2.4) + 0.70 \cdot (5 - 3.8)}{0.85 + 0.70} \right) = 4.87$$

<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>
Improving the Similarity / Prediction Functions

- **Neighborhood selection**
  - use similarity threshold instead of fixed number of neighbors

- **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

- **Number of co-rated Items**
  - Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
Memory-based and Model-based Approaches

- **User-based CF is said to be "memory-based"**
  - The rating matrix is directly used to find neighbors and make predictions
  - To predict we compute user similarity online and collect the ratings of the most similar ones. Such a KNN approach is called lazy learning.
  - This **does not scale** for large e-commerce sites, which have millions of customers

- **Model-based approaches**
  - We build a model offline
  - We use the model we computed **offline** to make predictions **online**
  - Models are updated / re-trained periodically
  - Examples
    1. Item-based collaborative filtering
    2. Probabilistic methods
    3. Matrix factorization
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**
  - similarity metric to find similar items which focuses on non-zero rating pairs
  
  \[
  \text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \ast |\vec{b}|}
  \]
  
  - cosine similarity does not take the differences in the average rating behaviour of different users into account

- **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:
  Weight ratings by item similarity

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

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

\( r_{ui} \): Alice’s rating for items i

\( \text{sim}(i, p) \): Similarity of item i with target item p

- No need to adjust rating scale as we only use ratings by Alice
Offline Pre-Calculations 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 $k$
  - An analysis of the MovieLens dataset indicates a $k$ 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 $k$ items above minimum similarity threshold
Collaborative Filtering Discussion

**Pros:**
- well-understood, works well in some domains
- 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 (see hybrid recommendation)
Literature