Recommender Systems

- Recommender systems suggest products to users that might be of interest to them.
- They help customers sift through the product catalogue and discover new products.

Customers who bought this item also bought:

- **AmazonBasics Mid-Back Office Chair, Black**
  - #1 Best Seller in Managerial Chairs & Executive...
  - $64.99 🌟prime

- **Furmax Office Chair PU Leather Gaming Chair, High Back Ergonomic Racing Chair, Desk Chair...**
  - #1 Best Seller in Office Drafting Chairs
  - $66

- **BestOffice High-back Computer Racing Gaming Chair**
  - 🌟🌟🌟🌟🌟 $89

- **Ergonomic Mesh Computer Office Desk Midback Task Chair w/Metal Base, One Pack**
  - 🌟🌟🌟🌟🌟 $52

- **Devoko Mid-Back Computer Office Swivel Desk Chair With Arms Height Adjustable...**
  - 💫 $39
Case Study

- Spanish retailer in the home décor business
- > 100 stores and an online shop
- In total 2.2 million transactions and 800,000 members of a loyalty programme

- Female: 90%
- Male: 10%

Age distribution:

- Female: 90%
- Male: 10%
User-Item Matrix

- **State-of-the-art**: analyse what ‘like-minded’ users enjoy
- Often based on explicit feedback (e.g. movie ratings)
- In retailing, interest needs to be inferred from implicit feedback, e.g. purchase history
- Preferences are stored in a large but sparse **User-Item matrix**

\[
\begin{array}{ccccc}
 & i_1 & i_2 & i_3 & i_4 & i_5 \\
 u_1 & 3 & 4 & 4 & \\
 u_2 & 1 & 2 & 3 & \\
 u_3 & 5 & & & \\
 u_4 & 1 & 3 & 2 & 3 \\
\end{array}
\]
1. Omni-Channel
Omni-Channel

- Using the **purchase history works well** as an indication of interest
- But there is much more information available to infer preference
- **Online Search** and **Surf Behaviour, Newsletters** and **Social Media**
Website – Searches

For confidentiality reasons, the example shows an unrelated company.
Weblogs – Searches

- Use the **search behaviour on the website** as an indicator for interest
- Analyse the stored HTML-GET requests for search queries (`q=`)
- Match the query to product name, description or product family
- Use cookies to identify users after they logged in once

**USER SEARCH:** red towel  
**SERVER LOG:** GET/ www.url.com/q=red towel
Website – Visits

* for confidentiality reasons, the example shows an unrelated company
Weblogs – Visits

▶ Use the **website visits** as an indication of interest
▶ Analyse the stored HTML-GET requests for product pages
▶ Use cookies to identify users after they logged in once

**VISITED CATEGORY:** bags  **SERVER LOG:** GET/ www.url.com/bags.html
Company sends out newsletters for new products and discounts.

Record newsletter opens and click throughs as an indication of preference.

Identify users based on personalised link / cookie.

* for confidentiality reasons, the example shows an unrelated company
Social Media

- Company posts promotions to Facebook.
- Use Facebook comments and likes as an indication of interest.
- Identify users based on name.

* for confidentiality reasons, the example shows an unrelated company.
A Recommender System Based on Omni-Channel Customer Data

Omni-Channel

Purchase History

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Weblog

<table>
<thead>
<tr>
<th></th>
<th>2s</th>
<th>1v</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1v</td>
<td>2s</td>
</tr>
<tr>
<td>1</td>
<td>1v</td>
<td></td>
</tr>
</tbody>
</table>

Social Media

<table>
<thead>
<tr>
<th></th>
<th>1l</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1p</td>
<td>1l</td>
<td></td>
</tr>
</tbody>
</table>

Newsletter

<table>
<thead>
<tr>
<th></th>
<th>1o</th>
<th>1c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1v</td>
<td>1c</td>
<td>2o</td>
</tr>
</tbody>
</table>

Preference Matrix

<table>
<thead>
<tr>
<th></th>
<th>u1</th>
<th>u2</th>
<th>u3</th>
<th>u4</th>
</tr>
</thead>
<tbody>
<tr>
<td>i1</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>i2</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>i3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>i4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Recommender System Based on Omni-Channel Customer Data

Information per Channel

- **Purchases**: 643,409
- **Website Searches**: 48,127
- **Website Visits**: 2,410,111
- **Newsletter Open**: 544,075
- **Newsletter Click**: 28,359
- **Facebook**: 396,662

Matthias Carnein

Omni-Channel

Recommendations

Evaluation

Conclusion
Store Location

- Detect WiFi probe requests from phones
- Identify users based on purchases with MAC address present
- **Possible Scenario**: Trigger a recommendation when customer visited a store but left without a purchase
- Because of **privacy concerns**, we did not make use of this data
2. Recommendations
**k-Nearest Neighbour (kNN)**

- Weighted average of ‘like-minded’ users [Liu 2011]
- Improvement: Weighted average over items

User-based CF:

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
<th>$i_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_2$</td>
<td>1</td>
<td>?</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$u_3$</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_4$</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

$r_{2,3} = 3$

Item-based CF:

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
<th>$i_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_2$</td>
<td>1</td>
<td>?</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$u_3$</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_4$</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

$r_{2,3} = 3$
Singular Value Decomposition (SVD)

- Split the large User-Item matrix in two smaller matrices [Funk 2018]
- By detecting hidden characteristics that approximate the purchase behaviour
- Special case for implicit feedback using Alternating Least Squares [Hu et al. 2008]
Co-Clustering

- Cluster users and items such that prediction error is minimized [George et al. 2005]
- Implicitly forms co-clusters of users and items
- Use the average rating of user, item, and User-Item cluster for prediction
Baseline

- For comparison we include a baseline algorithm
- Randomly recommends by sampling from the User-Item matrix

\[
\begin{array}{c|ccccc}
& i_1 & i_2 & i_3 & i_4 & i_5 \\
\hline
u_1 & 3 & 0 & 4 & 0 & 4 \\
u_2 & 1 & 0 & ? & 2 & 3 \\
u_3 & 0 & 5 & 0 & 0 & 0 \\
u_4 & 1 & 0 & 3 & 2 & 3 \\
\end{array}
\]

\[\hat{r}_{2,3} = \text{random}(r_{u,i})\]
3. Evaluation
### Masking

**User-Item Matrix**

<table>
<thead>
<tr>
<th>4</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Fill Zero & Binarize**

<table>
<thead>
<tr>
<th>1 0 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1</td>
</tr>
<tr>
<td>0 1 0 1 0</td>
</tr>
<tr>
<td>1 0 1 0 1</td>
</tr>
</tbody>
</table>

**Train set**

<table>
<thead>
<tr>
<th>1 0 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 1</td>
</tr>
<tr>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td>1 0 1 0 0</td>
</tr>
</tbody>
</table>

**Test set**

<table>
<thead>
<tr>
<th>1 0 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1</td>
</tr>
<tr>
<td>0 1 0 1 0</td>
</tr>
<tr>
<td>1 0 1 0 1</td>
</tr>
</tbody>
</table>
## Rank Measure

- **Ideal:** Relevant products should be ranked higher
- **Measure:** Average percentile-rank of items where the user has interest

<table>
<thead>
<tr>
<th>Product</th>
<th>Percentile Rank</th>
<th>Purchase</th>
<th>Rank Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chandellier</td>
<td>0%</td>
<td>✓</td>
<td>0%</td>
</tr>
<tr>
<td>Blackout Curtain</td>
<td>3%</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Wall Clock</td>
<td>6%</td>
<td>✓</td>
<td>6%</td>
</tr>
<tr>
<td>Towel</td>
<td>97%</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Canvas Art</td>
<td>100%</td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>
Recommendation Quality

- SVD (ALS): 7.76
- SVD: 34.10
- Co-Clustering: 16.97
- Item-based CF: 11.11
- Baseline: 49.99
4. Conclusion
Conclusion

- Recommender System based on **Omni-Channel Customer Data**
- Using information from purchases, search and browsing behaviour as well as newsletter and Facebook interactions
- Implemented using a real-life case
- Very high recommendation accuracy

Outlook:

- Improve customer recognition across the different channels
- Observe recommendation performance in real-life
References

DOI: 10.1007/978-3-642-19460-3.


Available Data

- Purchases
- Weblog
- Facebook
- Newsletter
- WiFi-logs

**Training Time**

A Recommender System Based on Omni-Channel Customer Data

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD (ALS)</td>
<td>1.5</td>
</tr>
<tr>
<td>SVD</td>
<td>5,051.9</td>
</tr>
<tr>
<td>Co-Clustering</td>
<td>644.2</td>
</tr>
<tr>
<td>Item-based CF</td>
<td>7,855.2</td>
</tr>
<tr>
<td>Baseline</td>
<td>0</td>
</tr>
</tbody>
</table>

Matthias Carnein
Using data from all available channels considerably improves the number of ‘item ratings’ per user.

Particularly for users that can be identified across multiple channels.
String Matching

- **Requirement**: partial matching and independence of word order
- **Token-set ratio** [SeatGeek 2017]: split two strings $X$ and $Y$ into intersecting words and two remainders

$$
t_0 = \text{sort}(\text{intersection})
$$
$$
t_1 = \text{sort}(\text{intersection}) + \text{sort}(\text{remainder } X)
$$
$$
t_2 = \text{sort}(\text{intersection}) + \text{sort}(\text{remainder } Y)
$$

- Compare relative similarity of each pair $t_0$, $t_1$, $t_2$ and choose pair with maximum similarity
- **Idea**: Similarity is larger if $X$ makes up a larger portion of $Y$ or if the remainders are similar