2. Recommender Systems
Recommenders Everywhere
Recommenders Everywhere
Outline

2.1. What are Recommender Systems?

2.2. Collaborative Filtering

2.3. Content-Based Recommendation

2.4. Hybridization & Evaluation
1. What are Recommender Systems?

- Recommender systems are about matching users and items.

- Recommender systems are about discovery not search:
  - no explicit information need; no explicit query
  - rather: “entertain me”, “show me something interesting”

- Recommender systems have big business impact [5]
  - 66% of movies watched on Netflix have been recommended
  - 35% of sales of Amazon.com are based on recommendations
Goals

- **User**: A good recommender brings up items that are
  - **relevant** (i.e., the user likes them once he uses them)
  - **novel** (i.e., the user does not yet know about the items)
  - **surprising** (i.e., the items are different from what the user already knows)

- **Company**: A good recommender brings up items that
  - users are likely to purchase (i.e., buy, rent, watch)
  - have high margins (e.g., to drive earnings)
**Netflix Prize**

- **Competition** by Netflix video rental company
  - *driver* for research in recommender systems
  - ran over **three years** (2007 – 2009)
  - goal was to **beat CineMatch** (Netflix’s recommendation algorithm) **by more than 10%** in terms of **root mean squared error** (RMSE)
  - award: **$1,000,000**
  - included a **data release** (100M ratings from 480K users for 17K movies); now **retracted** due to legal issues
  - winning approach **BellKor’s Pragmatic Chaos** [2] was a combination of several independently proposed approaches
Different research communities (e.g., DM, IR, ML) have worked on recommender systems and come up with very different ideas.

- Collaborative filtering only assumes (partial) knowledge about how useful specific items are to specific users (e.g., ratings).

- Content-based recommendation, in addition, knows about properties of the items (e.g., cast of movie, content of book).

- Hybridization strategies aim to provide better recommendations by systematically combining multiple baseline recommenders.
2. Collaborative Filtering

- **Collaborative filtering** only assumes (partial) knowledge about how useful specific items are to specific users (e.g., ratings)

- **No background knowledge** about items (e.g., cast or content) or users (e.g., age, gender, location)

- **Challenges**:  
  - recommend few items from a large pool  
  - data sparsity (large number of users and items)  
  - scalability
Explicit vs. Implicit Utility

- **Explicit utility values** are directly provided by users (e.g., ratings)
  - none available for new users (cold start problem)
  - users are **typically reluctant** to provide ratings
  - not necessarily comparable (pessimists vs. optimists)

- **Implicit utility values** can be obtained by observing users
  - based on **transactions** (e.g., purchases or clicks)
  - by **measuring engagement** (e.g., time spend watching video)
Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>4</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Utility Matrix

\[
r_{2,3} = 3
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Utility Matrix

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r2,3 = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$I_2 = \{1, 3, 4\}$
Utility Matrix

\[ r_{2,3} = 3 \]

\[ I_2 = \{1, 3, 4\} \]

\[ \bar{r}_2 = \frac{6}{3} = 2 \]
Utility Matrix

\[ r_{2,3} = 3 \]
\[ I_2 = \{1, 3, 4\} \]
\[ \bar{r}_2 = \frac{6}{3} = 2 \]
\[ U_2 = \{1, 5\} \]
Characteristics

- **Most values** of the utility matrix **are missing**, i.e., the **data is very sparse** (e.g., in Netflix dataset only 1% of values is known).

- **Missing values** are **different from zeros** and do not indicate that the user dislikes the item.

- **Magnitude of utility values** (e.g., ratings) differs from user to user (optimists vs. pessimists).
2.1. User-User Collaborative Filtering

- User-user collaborative filtering aka. *k-NN collaborative filtering* as first generation of recommenders (proposed in early 1990’s)

- **Idea**: Recommend items that are of high utility to similar users
2.1. User-User Collaborative Filtering

- User-user collaborative filtering aka. **k-NN collaborative filtering** as first generation of recommenders (proposed in early 1990’s)

- **Idea**: Recommend items that are of high utility to **similar users**
2.1. User-User Collaborative Filtering

- User-user collaborative filtering aka. **k-NN collaborative filtering** as first generation of recommenders (proposed in early 1990’s)

- **Idea**: Recommend items that are of high utility to **similar users**
2.1. User-User Collaborative Filtering

- User-user collaborative filtering aka. **k-NN collaborative filtering** as first generation of recommenders (proposed in early 1990’s)

- **Idea**: Recommend items that are of high utility to similar users

![Diagram showing user-user collaborative filtering and a matrix indicating user preferences](image-url)
2.1. User-User Collaborative Filtering

- User-user collaborative filtering aka. k-NN collaborative filtering as first generation of recommenders (proposed in early 1990’s)

- **Idea:** Recommend items that are of high utility to similar users
Measures of User Similarity

๏ How can we measure the similarity between two users $u$ and $v$?

๏ **Pearson correlation** (on items with known utility for both users)

$$s(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u) \cdot (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

๏ **Cosine similarity** (missing utility values as zeros)

$$s(u, v) = \frac{\sum_i (r_{u,i} \cdot r_{v,i})}{\sqrt{\sum_i r_{u,i}^2} \cdot \sqrt{\sum_i r_{v,i}^2}}$$
Generating Recommendations

- Identify neighborhood $N(u, k)$ of $k$ users most similar to $u$

- **Predict utility** of item $i$ as

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N(u, k)} s(u, v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u, k)} s(u, v)}$$

- Recommend $n$ items having **highest predicted utility**
Discussion

- **Pearson correlation** and **cosine similarity** only work if users \( u \) and \( v \) have **known utility values for common item** (e.g., have rated at least one common movie)

- User similarity is **sensitive to updates** (e.g., additional ratings) so that precomputing user similarities is not attractive

- **Neighborhood computation** is computationally expensive
2.2. Item-Item Collaborative Filtering

- **Item-item collaborative filtering** addresses the shortcomings of user-user collaborative filtering (proposed in early 2000’s)

- **Idea**: Recommend **items that are similar** to items of high utility
2.2. Item-Item Collaborative Filtering

- **Item-item collaborative filtering** addresses the shortcomings of user-user collaborative filtering (proposed in early 2000’s)

- **Idea**: Recommend **items that are similar** to items of high utility
2.2. Item-Item Collaborative Filtering

Item-item collaborative filtering addresses the shortcomings of user-user collaborative filtering (proposed in early 2000’s).

Idea: Recommend items that are similar to items of high utility.
Measures of Item Similarity

๏ How can we measure the similarity between two items $i$ and $j$?

๏ **Pearson correlation** (on users with known utility for both items)

$$ s(i, j) = \frac{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r}_u) \cdot (r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} (r_{u,j} - \bar{r}_u)^2}} $$

๏ **Cosine similarity** (missing utility values as zeros)

$$ s(i, j) = \frac{\sum_u (r_{u,i} \cdot r_{u,j})}{\sqrt{\sum_u r_{u,i}^2} \cdot \sqrt{\sum_u r_{u,j}^2}} $$
Generating Recommendations

- **Predict utility** of item \( i \) as

\[
\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{j \in S(u,i,k)} s(i,j) \cdot (r_{u,j} - \bar{r}_u)}{\sum_{j \in S(u,i,k)} s(i,j)}
\]

with \( S(u,i,k) \) as the set of \( k \) items with **known utility** for user \( u \) that are **most similar** to item \( i \)

- Recommend \( n \) items having **highest predicted utility**
Discussion

- **Pearson correlation** and **cosine similarity** only work if items i and j have **known utility values for common user** (e.g., have been rated by the same user).

- Item similarity is **less sensitive to updates** (e.g., additional ratings), assuming that there are **many more users than items**.

- In practice, **item similarities are typically precomputed**, and **truncated** (keeping top-k most similar items per item).
2.3. Association Rules

- **Association rule mining** developed for market basket analysis to learn rules (patterns) from customer transactions (e.g., buys soda and beer => buys snacks)

- Association rules can be used to generate recommendations by considering items with known utility per user a transaction

- Let A and B be set of items, we are interested in identifying association rules A => B with sufficient support and confidence
Support and Confidence

- For a set of items (itemset) $A$ its **support** $s(A)$ is the fraction of transactions that contains $A$

\[
s(A) = \frac{\# \text{ transactions containing } A}{\# \text{ transactions}}
\]

- For an association rule $A \Rightarrow B$ its **confidence** $c(A \Rightarrow B)$ is the fraction of transactions containing $A$ that also contain $B$

\[
c(A \Rightarrow B) = \frac{\# \text{ transactions containing } A \cup B}{\# \text{ transactions containing } A}
\]
Identifying Frequent Itemsets

- **Apriori algorithm** [1] can be used to identify frequent itemsets having a support above a minimum support threshold.

- **Iterative algorithm** exploiting anti-monotonicity of supports:

  \[ A \subset B \Rightarrow s(A) \geq s(B) \]

- **Sketch:**
  - **identify** frequent 1-itemsets (i.e., containing a single item)
  - **repeat** (until no frequent k-itemsets are found)
    - **generate candidates** by joining frequent (k-1)-itemsets
    - **prune** infrequent candidates and emit frequent k-itemsets
Generating Association Rules

- Generate **association rules** from frequent itemset $X$
  - consider every **non-empty subset** $A \subset X$ and let $B = X \setminus A$
  - output **association rule** $A \Rightarrow B$ if $c(A \Rightarrow B)$ above threshold
Generating Recommendations

- Consider all items $I_u$ with known utility for user $u$
  - identify all association rules $A \Rightarrow B$ so that $A \subseteq I_u$
  - items from $B \setminus I_u$ are candidates for recommendation; for each candidate keep track of highest confidence of any association rule suggesting it
  - recommend $n$ items having highest confidence
2.4. Dimensionality Reduction

- **Idea**: Identify a **small number** (in comparison to \( m \) and \( n \)) of **common interests** (topics) to represent users and items; recommend items to users that belong to the same topics.

- Utility matrix \( R \) can be seen as user vectors (in a \( m \)-dimensional vector space) or item vectors (in a \( n \)-dimensional vector space).

- Dimensionality reduction methods reveal the **latent structure** of a matrix by representing it as a **product of multiple smaller matrices** (e.g., UV decomposition, singular value decomposition, principal component analysis).
Singular Value Decomposition

- Determine **k-SVD of utility matrix** \( R \) (\( m \times n \))

\[
R \approx U^k \Sigma^k T^T
\]

as best possible rank-\( k \) approximation under Frobenius norm

- \( U \) captures **user-topic associations**
- \( \Sigma \) captures **topic importance**
- \( T \) captures **item-topic associations**
**Imputation**

- **SVD** requires a **complete matrix** but **R misses a lot of values**

- **Imputation** is the process of filling missing values with defaults
  - average utility assigned to item by different users
  - average utility assigned to other items by same user
  - other baseline predictors
Generating Recommendations

- **Predict utility** of item $i$ for user $u$ as

  \[
  \hat{r}_{u,i} = \sum_k U_{u,k} \cdot \Sigma_{k,k} \cdot T_{k,i}^T
  \]

- **Predict utilities** of all items for user $u$ as

  \[
  U_u \times \Sigma \times T^T
  \]
3. Content-Based Recommendation

- Content-based recommendation assumes (partial) knowledge about how useful specific items are to specific users and background knowledge about properties of the items.

- Idea: Recommend items that are similar to items of high utility.
3. Content-Based Recommendation

- **Content-based recommendation** assumes (partial) knowledge about how useful specific items are to specific users and background knowledge about properties of the items.

- **Idea**: Recommend items that are similar to items of high utility.
3. Content-Based Recommendation

- **Content-based recommendation** assumes (partial) knowledge about how useful specific items are to specific users and background knowledge about properties of the items.

- **Idea**: Recommend items that are similar to items of high utility.

![Movie posters representing content-based recommendations](image-url)
3. Content-Based Recommendation

- **Content-based recommendation** assumes (partial) knowledge about **how useful specific items are to specific users** and background knowledge about **properties of the items**

- **Idea:** Recommend **items that are similar** to items of high utility
Items and Users as Vectors

- Represent **items as vectors** in a high-dimensional vector space (works well, for instance, for text documents with tf.idf weighting)

- Represent **user as vector** obtained as **weighted combination of item vectors** of items with known utility values

\[
\vec{u} = \sum_{i \in I_u} \left( \frac{r_{u,i}}{\sum_{j \in I_u} r_{u,j}} \right) \cdot \vec{v}_i
\]

- Recommend items with **high cosine similarity** to user vector

\[
\begin{align*}
\vec{v}_{RoK} &= \begin{bmatrix} 0.13 \\ \vdots \\ 0.65 \end{bmatrix} \\
\vec{v}_{GC} &= \begin{bmatrix} 0.04 \\ \vdots \\ 0.55 \end{bmatrix}
\end{align*}
\]
Domain-Specific Item Similarity

- **Not all item properties** are suitable for representation in vector and we may lose their semantics when doing so.
- **Category** (e.g., /Travel/U.S.A., /Travel/Canada, /Cooking/Italian)
- **Year** (e.g., 1980 should be less similar to 2002 than 1981)

- Define **domain-specific item similarity** based on their properties, for instance, as weighted sum of property-specific similarities

\[
s(RoK, GC) = \alpha \cdot s_a(RoK, GC) + \beta \cdot s_y(RoK, GC) + \gamma \cdot s_c(RoK, GC)
\]
Domain-Specific Item Similarity

- Recommend **items that are similar** to items of high utility

\[
\text{score}(u, j) = \sum_{i \in I_u} r_{u,i} \cdot s(i, j)
\]
4. Hybridization & Evaluation

- **Combining different recommenders** can be attractive
  - improved recommendations (cf. winner of Netflix competition)
  - overcoming cold start problems
  - improved performance

- **Hybridization strategies** systematically combine recommenders
  - **Ensemble** (combine outputs of different recommenders)
  - **Switch** (choose recommender to use)
**Ensemble**

- Obtain (top-k) recommendations from **multiple recommenders**
- Combine **recommendations** by aggregating per item
  - **predicted utility** by different recommenders
  - **reciprocal rank** in output of different recommenders
  - **votes** (item in output) from different recommenders

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>R2</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.6</td>
<td>0.2</td>
<td>0.1</td>
<td></td>
<td></td>
<td>0.5</td>
<td>0.4</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>utility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0.4</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>1/rank</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1/1</td>
<td>4/3</td>
<td>1/2</td>
<td>1/3</td>
<td>1/2</td>
</tr>
<tr>
<td></td>
<td>vote</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Switch

- **Decide** (or learn to decide) when to use **which recommender**

- **Example**: Collaborative filtering suffers from **cold start problem**
  - use content-based recommender, if user has too few known utility values (e.g., has rated too few items)
  - otherwise, use item-item collaborative filtering
Evaluation

- Recommender systems can be evaluated like other IR systems
  - user judges whether recommended items are relevant
  - determine precision, recall, F1
  - captures only whether relevant items are returned

- More commonly, the focus is on prediction accuracy
  - split utility values from dataset (e.g., movie ratings) into training and test data (repeat multiple times)
  - measure mean absolute absolute error on test data

\[
\frac{1}{n} \sum_{(u,i)} |\hat{r}_{u,i} - r_{u,i}|
\]
Summary

- **Recommender systems** help users to discover relevant and surprising items and drive many of today’s businesses.

- **Collaborative filtering** uses only knowledge about how useful items are to users; variety of approaches have been proposed.

- **Content-based recommendation** also uses knowledge about properties of the items (e.g., content); IR-style approaches.

- **Hybridization strategies** combine multiple recommenders, for instance, to obtain better recommendations or performance.

- **Evaluation of recommender systems** usually focuses on prediction accuracy and uses training/test splitting of data.
When Recommender Systems Fail

References

VLDB 1994

Collaborative Filtering Recommender Systems,

[3] Y. Kohen: The BellKor Solution to the Netflix Grand Prize
http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

Available at: http://www.mmds.org

Tutorial at European Summer School for Information Retrieval, 2013