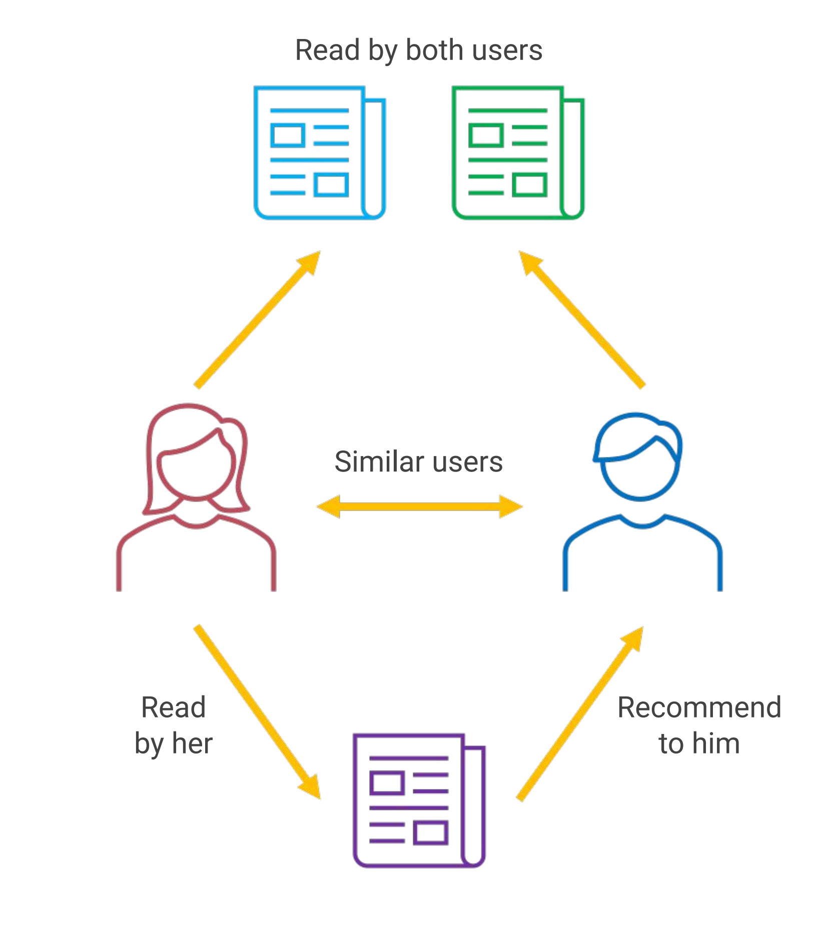
Recommender systems

Alaa BAKHTI

Collaborative Filtering (CF)

Predict the user interests by collecting preferences or taste information from many users.



Different types of collaborative filtering

Memory-based CF

- Find correlations (similarities) between users or items using the users rating data.
- Rely on simple similarity measures to match similar people or items together.

Model-based CF

- Recommendation as an optimisation problem: use machine learning algorithms to predict ratings.

Give an example of ML model that is Memory-based and another one that is Model-based.

Similarity metrics

Utility matrix

	I1	I2	I3	I4	I5	I6	I7
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

The user A is more similar to user B, C or D? Why?

Find the similarity metric *sim(X, Y)* that captures the intuition that *sim(A, B) > sim(A, C)*

Jaccard similarity

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$\text{sim}(A,B) = \frac{r_A \cap r_B}{r_A \cup r_B} = \frac{1}{5} = 0.2$$

$$\text{sim}(A,C) = \frac{2}{4} = 0.5$$

$$\text{sim}(A,B) < sim(A, C)$$

	I1	I2	I3	I4	I5	I6	I7
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Problem : *The Jaccard similarity ignores rating values*

Cosine similarity

$$sim(A, B) = \cos(A, B) = \frac{A \cdot B}{\| A \| \, \| B \|}$$

$$sim(A, B) = 0.38$$

$$sim(A, C) = 0.32$$

$$sim(A, B) > sim(A, C)$$

	I1	I2	I3	I4	I5	I6	I7
A	4	0	0	5	1	0	0
B	5	5	4	0	0	0	0
C	0	0	0	2	4	5	0
D	0	3	0	0	0	0	3

No big difference between sim(A, B) and sim(A, C)

Problem : *Treats missing ratings as negative ratings*

Centered cosine similarity

$$sim(A, B) = \cos(A - \bar{A}, B - \bar{B})$$

- Normalize ratings by subtracting user rating mean.
- 0 becomes the average rating of the user
  - Positive ratings : ratings > 0
  - Negative ratings : ratings < 0

	I1	I2	I3	I4	I5	I6	I7
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

$$sim(A, C) = -0.56$$

$$sim(A, B) = 0.09$$

$$sim(A, B) > sim(A, C)$$

- → Missing ratings treated as average
- → Handles “tough raters” and “easy raters”
- → Also called Pearson correlation

Memory-based CF

User-User

- Compare people based on their past behavior
- Find similar users based on their ratings of items.
- Recommend items that a user a watched to the user b if a & b are similar.

Limitations

- People tastes change
- Scalability
- Cold start problem on users and items

	I1	I2	I3	I4	I5	I6	I7
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Goal

Given:

- a user x
- A list of movies M that the user didn't watch

For each movie m from the set M

We want to predict the rating of the user x to the movie m

Algorithm : predict the rating of the movie m by the user x

- Take the subset of users *U* that rated the movie *m*
- Compute the similarity between the user *x* and all the users in *U*
- Select the neighbourhood *N* of users that are very similar to the user *x*
  - *top-k* neighbours or *sim(x, Ui)* > threshold
- Compute the rating of the movie *m* by user *x* using the ratings of the set *N* and the similarity of each user with the user *x*
  - Average prediction :

$$r_{xm} = \frac{1}{k} \sum_{y \in N} r_{ym}$$

- Problem of average prediction : highly and low similar users have the same weight
- Weighted average prediction : weight the ratings by the similarity of the user to the user *x*

$$r_{xm} = \frac{\sum_{y \in N} s_{xy} r_{ym}}{\sum_{y \in N} s_{xy}} \quad where\ s_{xy} = sim(x, y)$$

Item-Item

- Try to find similar movies based on users ratings.
- Was invented and used by amazon in 1998 [source][paper 2001].
  - Who viewed also viewed

Advantages

- Items does not change over time.
- Less computation to do than user-based approach: compute similarity between the movie m and the list of movies the user watched and not all the movies

	I1	I2	I3	I4	I5	I6	I7
A	4			5	1		
B	5	5	4	5			
C				2	4	5	
D		3					3

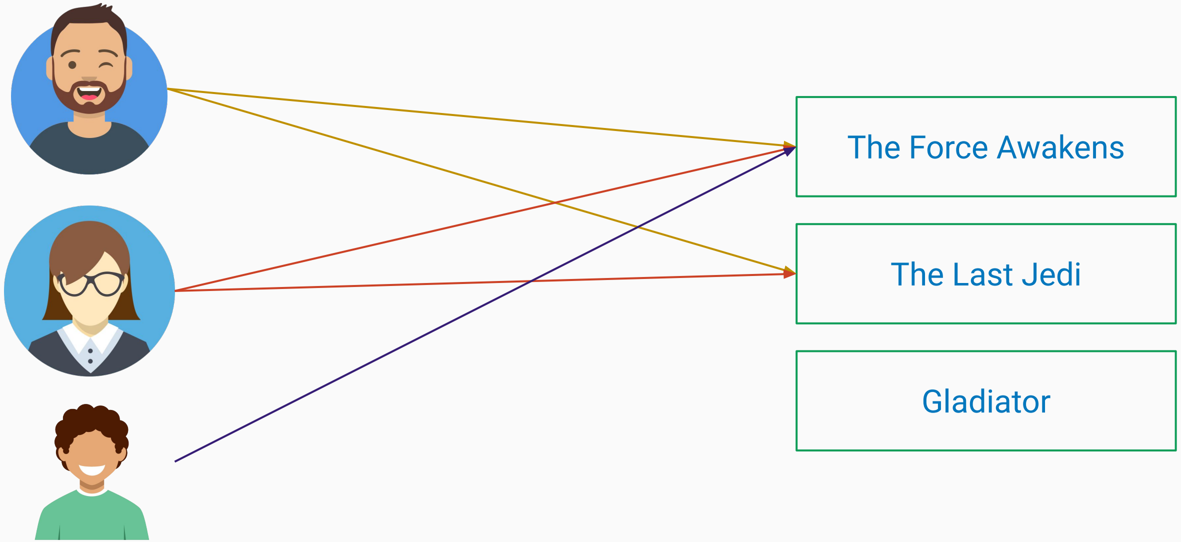
Algorithm 1 : predict the rating of the movie m by the user x

- Compute the similarity between the movie m and all the movies rated by the user x.
- Select the neighbourhood L of movies that are very similar to m
    - top-k neighbours or sim(Lj, m) > threshold).
- Use the *weighted average prediction* to compute the rating of the movie m by user x using the ratings of the set L and the similarity of each movie with m.

$$r_{xm} = \frac{\sum_{l \in L} s_{lm} r_{xl}}{\sum_{l \in L} s_{lm}} \quad where\ s_{lm} = sim(l,\ m)\ \ and\ r_{xl} = rating\ of\ \ user\ x\ on\ item\ l$$

Algorithm 2

- For each pair of movies, find the list of users that watched both of them.
- Create a profile for each movie using these user ratings.
- Determine if the 2 movies are similar based on their profiles similarity.
- 2 movies are similar if the users rated them the same way.



Practical work

In this practical work, you need to benchmark the following algorithms on the model evaluation using RMSE & MAE and the prediction execution time:

- User based algorithm
- Item based algorithm
- For each algorithm, you should use at least 2 similarity measures (cosine similarity, pearson correlation, …)

Instructions

- Use the cleaned dataset for the benchmarking
- To display your benchmarking results, you need to use charts (bar charts, etc)
- For the algorithms, you should use the surprise package and not implement them from scratch