Instituto Andaluz Interuniversitario en
**Data Science and Computational Intelligence**

DaSCI

UNIVERSIDAD DE GRANADA

SIGIR 2020

XI'AN·CHINA

# Reciprocal Recommendation

## Matching Users with the Right Users

Decision Support and Recommender Systems

*ACM SIGIR'20 Tutorial* *Iván Palomares Carrascosa* *ivanpc@ugr.es*

# About the presenter

**Iván Palomares Carrascosa**
DaSCI Research Institute, Granada, Spain

Research interests:

- **Recommender Systems**
- **Group and multi-criteria decision-making**
- **Preference modelling and fusion**
- **Applications**: participatory decisions, group recommendation, social recommendation, tourism, health.

Website: [www.ivanpc.com](www.ivanpc.com)

# Tutorial Materials

**Decision Support and Recommender Systems Lab website:**

http://dsrs-lab.ugr.es

Go to *'Events'* → *'ACM SIGIR'20 Tutorial'*

- Slides (PDF)
- Paper: *Reciprocal Recommender Systems: Analysis of State-of-the-Art Literature, Challenges and Opportunities on Social Recommendation.*

# Contents

**PART I**

- Reciprocal Recommender Systems (RRS)
- The Reciprocal Recommendation process I: ***prediction***
- The Reciprocal Recommendation process II: ***fusion***

**Virtual coffee break**

**PART II**

- Applications of RRS
- Evaluating RRS
- Challenges and future directions
- Discussion activity

# Reciprocal Recommender Systems (RRS)

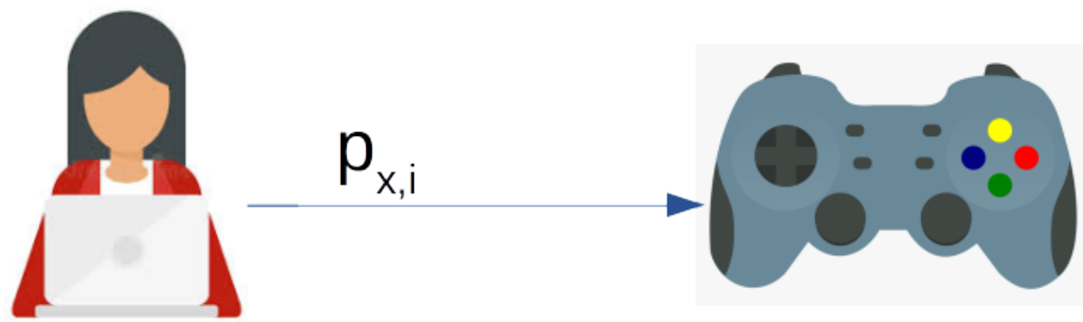
**Recommender Systems (RS)**

- Provide users with items (products, contents, …) they might be interested in, by analyzing their preferences, needs or behavior.
- Solution for the *information overload* problem in Internet sites.
- Applications: e-commerce, entertainment, retail, tourism, …

**What does a RS do?**

- Predict user-item preferences: how much a user may like an unseen item by him/her.
- Recommend the items predicted as most liked by the user.
- As the user interacts with the system, the system builds insight of the user's preferences.

# Reciprocal Recommender Systems (RRS)



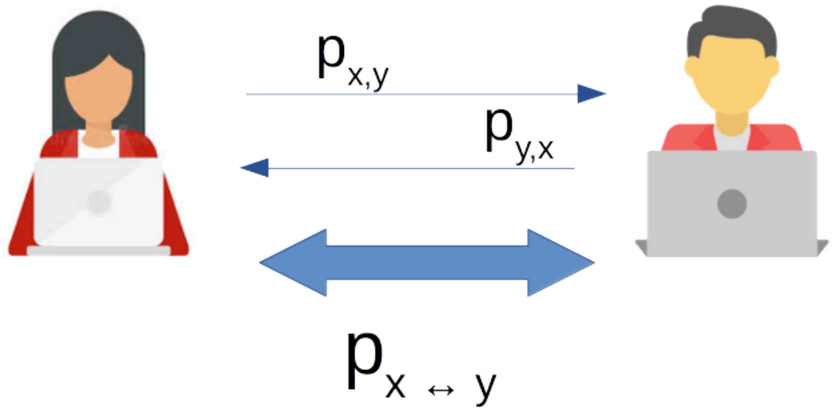

**Formal RS definition:** Given a user $x \in U$, a recommender $\mathcal{R}(x)$ is a system that recommends a list of items $R \subset I$ such that the degree of preference $p_{x,i}$ by $x$ towards every item $i \in R$ is stronger than the preference degree by $x$ towards any item $i' \notin R$:

$$\mathcal{R}_I(x) = \{i : p_{x,i} > p_{x,i'}, \forall i \in R, \forall i' \notin R\}$$

# Reciprocal Recommender Systems (RRS)

**Reciprocal Recommender Systems**

- Recommend users to each other.
- In their simplest form, they suggest *pairs* of users to connect.




- Both users must *reciprocate* – both of them should be satisfied with the "matching" suggestion.
- ***Bidirectional preference*** → determine mutual compatibility between users, e.g. reciprocal preference score $p_{x \leftrightarrow y}$

# Reciprocal Recommender Systems (RRS)

## From traditional item-to-user to reciprocal recommendation

| **Item-to-user recommender** | **Reciprocal Recommender** |
|---|---|
| Recommendations for $x \in U$ are items $i \in I$ | Recommendations for $x \in U$ are users $y \in U$ |
| Items can be usually recommended to multiple users | In some applications, if both $x$ and $y$ accept the matching recommendation, then they are no longer available for being recommended to anyone else. |
| A successful recommendation does not imply that the user leaves the system | In some applications, users may not need using the system and leave it after a successful recommendation |
| Success is determined by the user receiving the recommendation | Both $x$ and $y$ must be satisfied with the recommendation to deem it as successful |

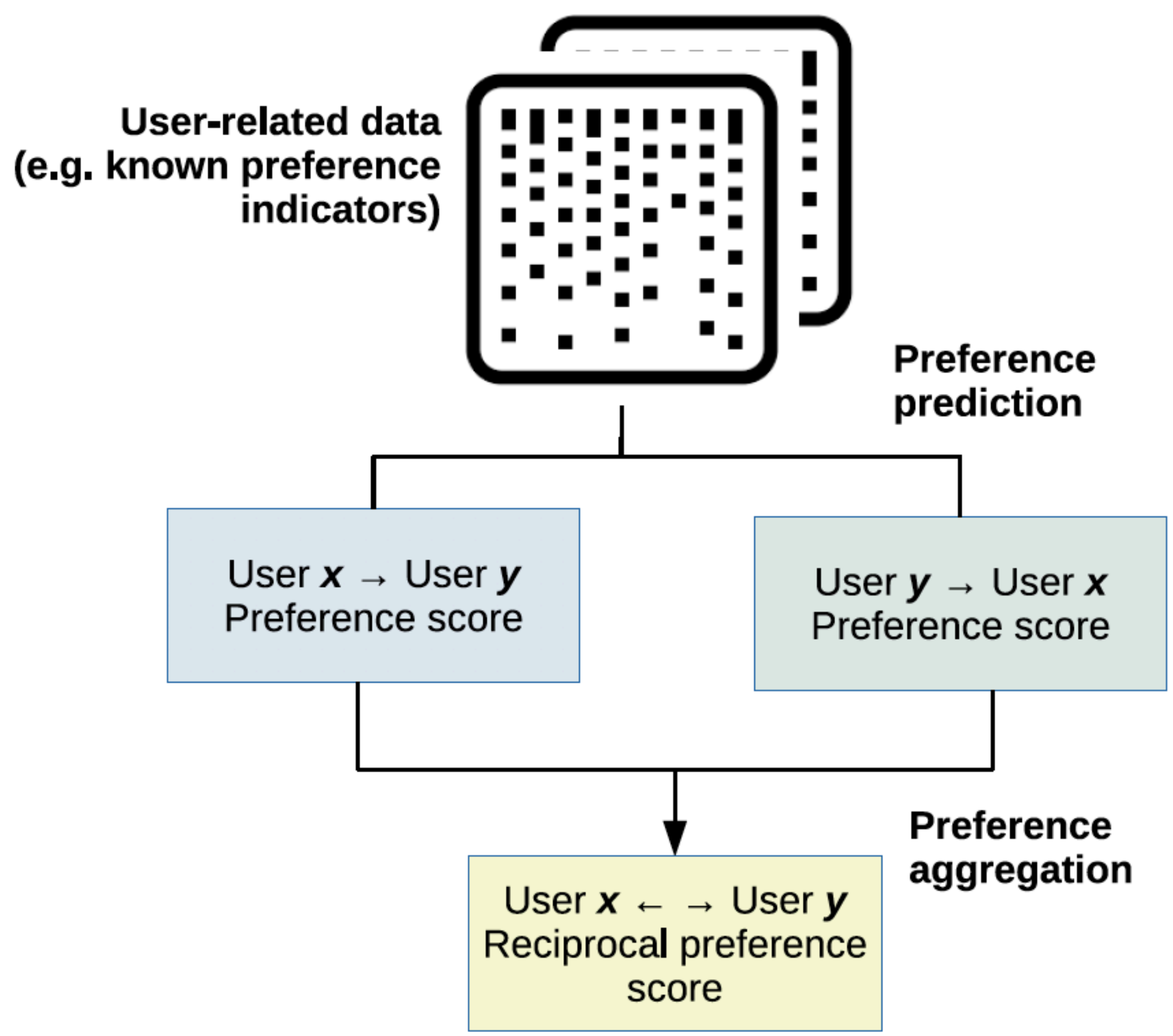
# Reciprocal Recommender Systems (RRS)

## Single-class *vs* two-class RRS

| **Single-class RRS** |
| --- |
| - Homogeneous set of users $U$<br>- Any two users in $U$ can be recommended to each other<br>- Applications: symmetric social networks, matching learners, homosexual dating, shared-economy, … |
| **Two-class RRS** |
| - $U$ subdivided in two disjoint subsets of users or classes, e.g. $U = M \cup F$<br>- Given $x \in M$, only users in the other class can be recommended, $y \in F$<br>- Applications: heterosexual dating, recruitment, student-mentor matching, … |

# Reciprocal Recommender Systems (RRS)

## General RRS conceptual model



# The reciprocal recommendation process I: Prediction

# Content-based Filtering (CB)

Rely on content features or item metadata to **recommend *similar* content to what I liked.**


How do CB principles apply in reciprocal recommendation?

# CB in Reciprocal Recommendation

- Users typically have ***profiles*** with information about themselves.
- They may also have information about types of users they are interested in.
- **Explicit preferences**: attributes of users sought, ratings, …
- **Implicit preferences**: they are learnt from user activity in the system: EoIs, viewed users' profiles, messaging, etc.

Given a subject user *x*, in a **CB-RRS**, knowledge is built about the type/attributes of users liked by *x*, i.e. her preferences.
Then a user *y* whose profile aligns with *x*'s preferences, is likely to be recommended to *x* …. (or is he?)

# CB in Reciprocal Recommendation

- Users typically have ***profiles*** with information about themselves.
- They may also have information about type of users they are interested in.
- **Explicit preferences**: attributes of users sought, ratings, …
- **Implicit preferences**: they are learnt from user activity in the system: EoIs, viewed users' profiles, messaging, etc.

Given a subject user *x*, in a **CB-RRS**, knowledge is built about the type/attributes of items (other users) liked by *x*, i.e. her preferences. A user *y* whose profile matches *x*'s preferences, is likely to be recommended to *x* ***iff x's profile also matches y's preferences.***

**Reciprocity!**

# RECON: A content-based RRS

**RECON** utilizes profile information and preferences inferred from messaging behavior to make reciprocal recommendations.

L. Pizzato, Pizzato, L., Rej, T., Chung, T., Koprinska, I., & Kay, J. (2010). *RECON: A reciprocal recommender for online dating*. Proceedings ACM RecSys '10, pp. 207-214.

***Basic Notation***

- *A*: Set of profile attributes: gender, age, height, eye color, …
- *x:* the user for whom recommendations will be produced.

**Subject user = recommendee = active user = target user**

- $U_x$: Profile of user $x$

$$U_x = \{v_{x,a} : \forall attributes\ a \in A\}$$

- $v_{x,a}$: Value of attribute $a \in A$ for user $x$.

# RECON: A content-based RRS

***Basic Notation***

- $M_{x,*}$: Users $x$ sent an EoI to (e.g. a message).
  - $M^{+}_{x,*}$: Users $x$ sent an EoI to **and responded positively (reciprocated).**
  - $M^{-}_{x,*}$: Users $x$ sent an EoI to **and responded negatively (rejected).**

- $M_{*,x}$: Users who sent an EoI to $x$.
  - $M^{+}_{*,x}$: Users who sent an EoI to $x$ **and $x$ responded positively (reciprocated).**
  - $M^{-}_{*,x}$: Users $x$ sent an EoI to **and $x$ responded negatively (rejected).**

- $R_x$: List of recommended users $y$ for $x$. It holds $R_x \cap \left(M_{x,*} \cup M_{*,x}\right) = \emptyset$

# RECON: A content-based RRS

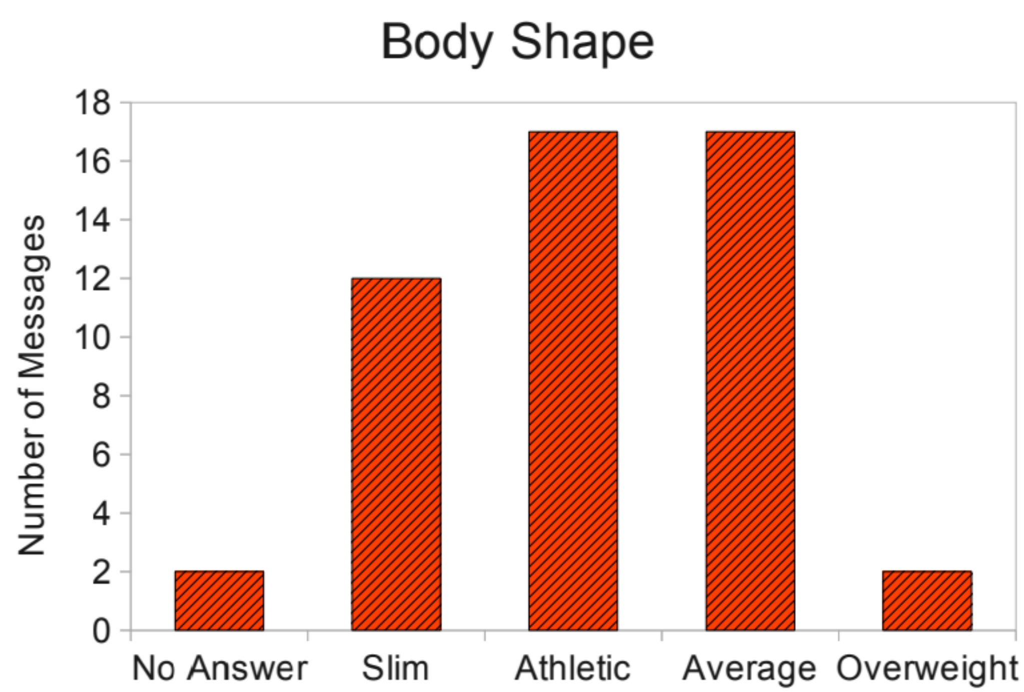
## *RECON key ideas (I) – building a preference model for x*

- For every $x$ and profile attribute $a \in A$, a **frequency distribution** is built, representing the **preferences** $P_x$ of $x$ on values of that attribute.
  - This is done by looking at occurrences of attributes' values in (1) users contacted by $x$, and (2) users to whom $x$ replied positively.

**$a$ = body shape**

$M_{x,*}$

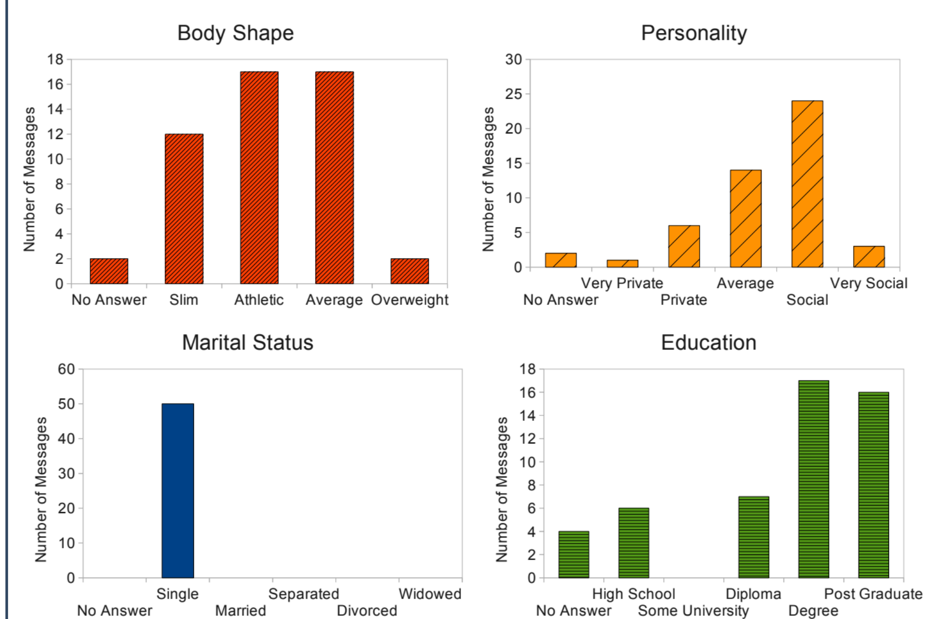
$M^{+}_{*,x}$



# RECON: A content-based RRS

***RECON key ideas (I) – building a preference model for x***

- Frequency distributions are built for *discrete* attributes.




Images source: Pizzato et al. (2010)

- But, how about **continuous attributes**, e.g. height and weight?
  - Not considered in the original model, but the *average value* exhibited by all users in $M_{x,*} \cup M^{+}_{*,x}$ could be taken.

# RECON: A content-based RRS

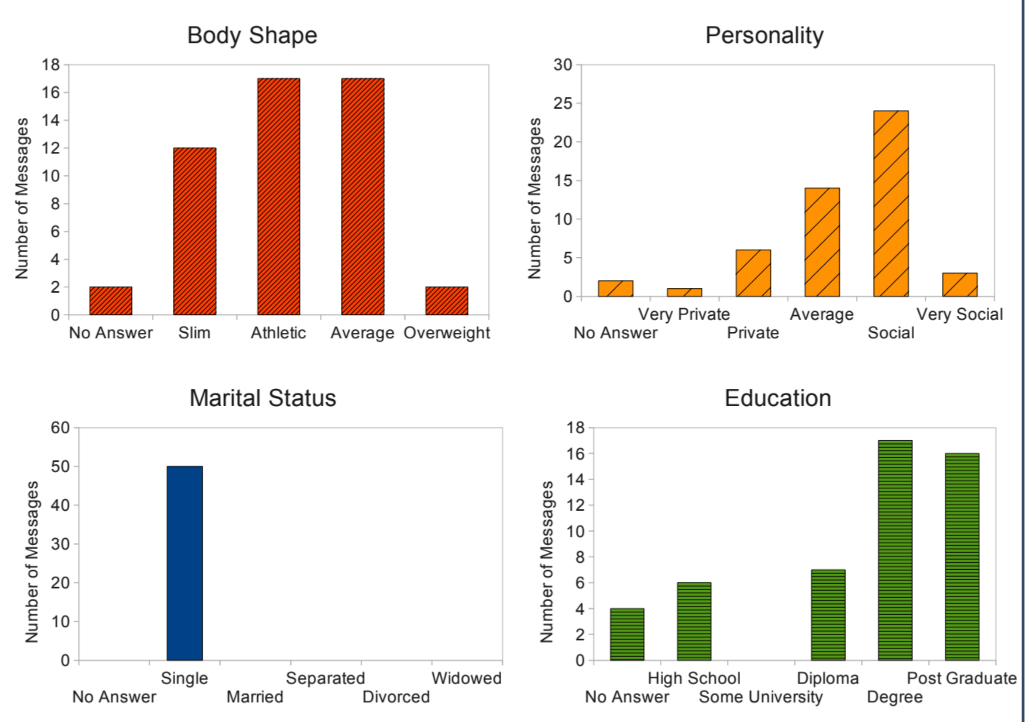
***RECON key ideas (II) – calculate compatibility with unknown users***

- $U_y$: User $y$ profile, such that $U_y = \{v_{y,a} : \forall attributes\ a \in A\}$
- For discrete attributes, add the frequencies in $P_x$ associated to $v_{y,a}$

<u>Example with four discrete attributes:</u>
User $y$ is 1.70cm tall, slim, social, single and has high-school education.

$s = 0.25 + 0.4 + 1 + 0.15 = \mathbf{1.8}$ *(out of 4)*

Then the compatibility or preference by $x$ towards $y$ is $p_{x,y} = 1.8/4 = \mathbf{0.45}$




Images source: Pizzato et al. (2010)

# RECON: A content-based RRS

*RECON key ideas (II) – calculate compatibility with unknown users*

- The $p_{x,y}$ values like the one we just calculated would be enough to produce a top-$N$ recommendation list … if we only needed to consider user $x$'s preferences.
- But we also need to account for $y$'s preferences → **Reciprocity!**

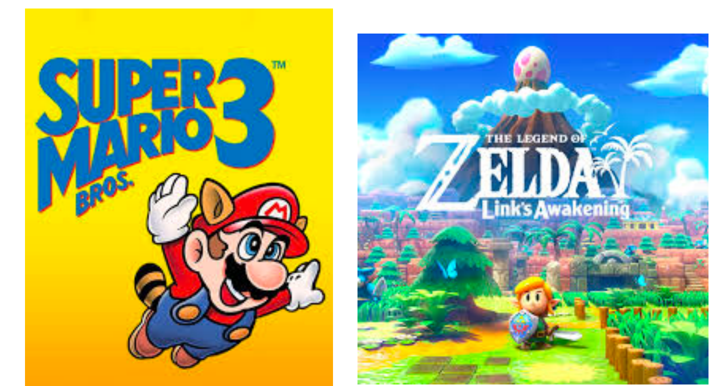
**Let's go the other way round:**

- Assume that for $P_y$ and $U_x$ we now get $p_{y,x} =$ **0.6.**
- We then **aggregate** both preference scores to get the reciprocal preference score of *mutual preference* between $x$ and $y$.

**Harmonic mean *IN ACTION*:** $\boldsymbol{p_{x\leftrightarrow y} = \frac{2}{0.45^{-1}+0.6^{-1}} = 0.514}$

# Collaborative Filtering (CF)

Recommend items liked by *similar* **users to me.**




How do CF principles apply in reciprocal recommendation?

# CF in Reciprocal Recommendation

- **CB-RRS** accounted for similarity between the subject users' preferences and unknown object users (treated as "items").
- In **CF-RRS** we observe the preferences of users who *interact* similarly (they liked similar object users) as the target user.
- User profile information is less relevant in CF, now we concentrate on analyzing user-user activity patterns, and identifying users with similar activity to us → neighbor users.

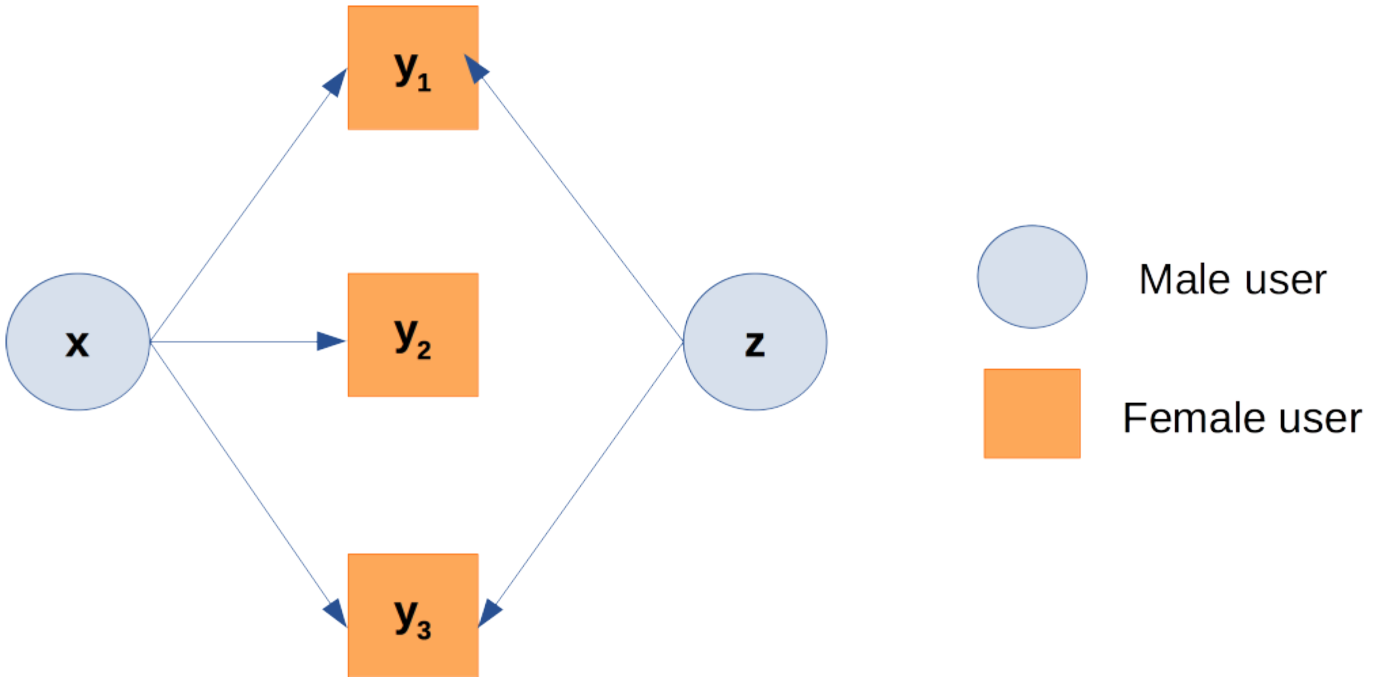
Given a subject user $x$, a **CF-RRS** analyses users' interactions with other users to identify users with similar interactions to $x$.

# CF in Reciprocal Recommendation

***User taste and user attractiveness***

**(1) Taste – determined by active user interaction:**

- If two users $x,z$ in the same class initiate positive interactions with several users $y_1$, $y_2$, $y_n$ in common, $x$ and $z$ have similar taste.

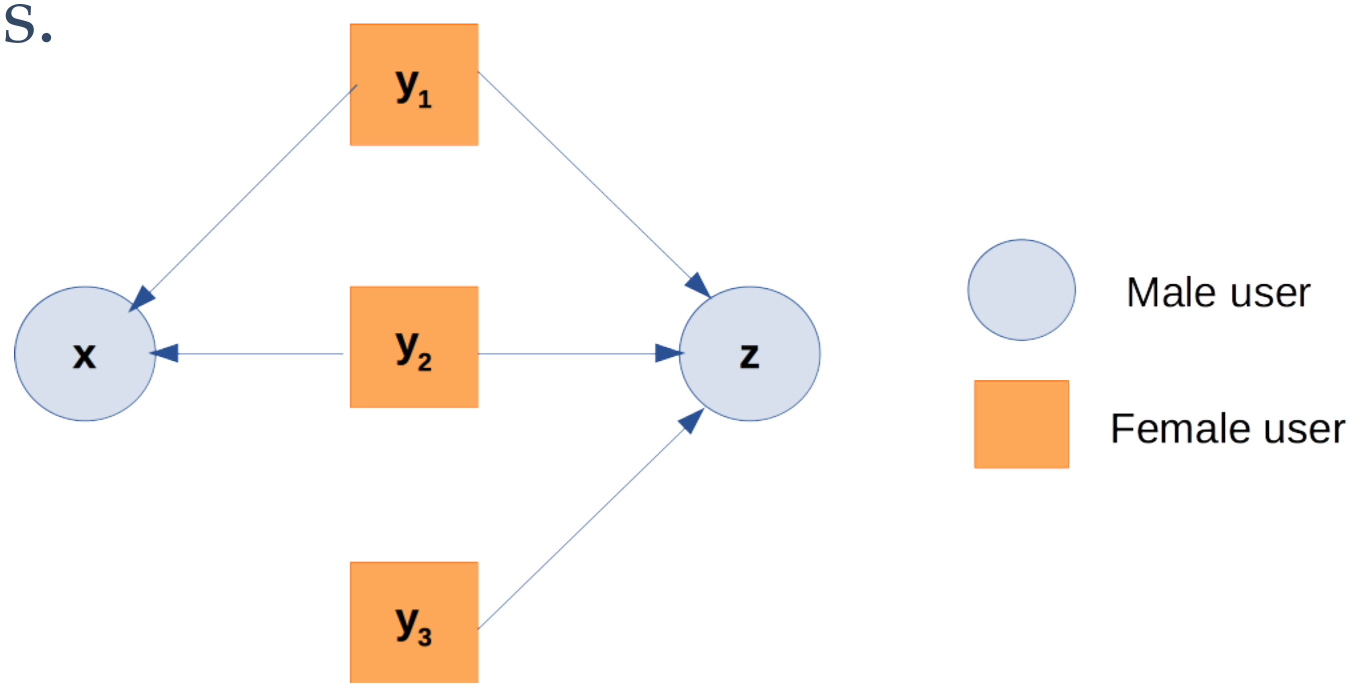


# CF in Reciprocal Recommendation

*User taste and user attractiveness*

**(2) Attractiveness – determined by passive user interaction:**

- If two users $x,z$ in the same class receive positive interactions from various other users $y_1$, $y_2$, $y_n$ in common, $x$ and $z$ have similar attractiveness.



# CF in Reciprocal Recommendation

### *CF assumptions in Reciprocal Recommendation*

If people with similar taste to $x$ like $y$, $x$ will like $y$.
If people with similar taste to $y$ like $x$, $y$ will like $x$.

If $x$ likes people with similar attractiveness to $y$, $x$ will like $y$.
If $y$ likes people with similar attractiveness to $x$, $y$ will like $x$.

***y* should be recommended to *x* when**: *y* likes people with similar attractiveness to *x* and *x* likes people with similar attractiveness to *y*,

**or equivalently**

when people with similar taste to *y* like *x* and people with similar taste to *x* like *y*.

X. Cai et al. (2010). *Collaborative Filtering for People to People Recommendation in Social Networks*. Proceedings AI 2010, pp. 476-485.

# RCF: A collaborative filtering RRS

**RCF** is a family of algorithms, some of them memory-based CF, relying on different *neighborhood* and *similarity* functions.

P. Xia, B. Liu, Y. Sun, C. Chen. (2015). *Reciprocal recommendation system for online dating*. Proceedings ASONAM'15, pp. 234-241.

### *Basic Ideas*

- Nearest-neighbour strategy
- The preference $p_{x,y}$ is determined based on the similarity between the **[taste / attractiveness]** of $x$ and users $z$ who positively interacted with $y$.
- **Interest similarity:**

$$sim(x,z) = \frac{EoI_{from}(x) \cap EoI_{from}(z)}{EoI_{from}(x) \cup EoI_{from}(z)}$$

**Attractiveness similarity:**

$$sim(x,z) = \frac{EoI_{to}(x) \cap EoI_{to}(z)}{EoI_{to}(x) \cup EoI_{to}(z)}$$

# RCF: A collaborative filtering RRS

- **Interest similarity:**

$$sim(x,z) = \frac{EoI_{from}(x) \cap EoI_{from}(z)}{EoI_{from}(x) \cup EoI_{from}(z)}$$

**Attractiveness similarity:**

$$sim(x,z) = \frac{EoI_{to}(x) \cap EoI_{to}(z)}{EoI_{to}(x) \cup EoI_{to}(z)}$$

- **Neighborhood function:** Users who had some interaction with $x$ (in the opposite class to $x$ in a two-class RRS modelled as a bipartite graph).

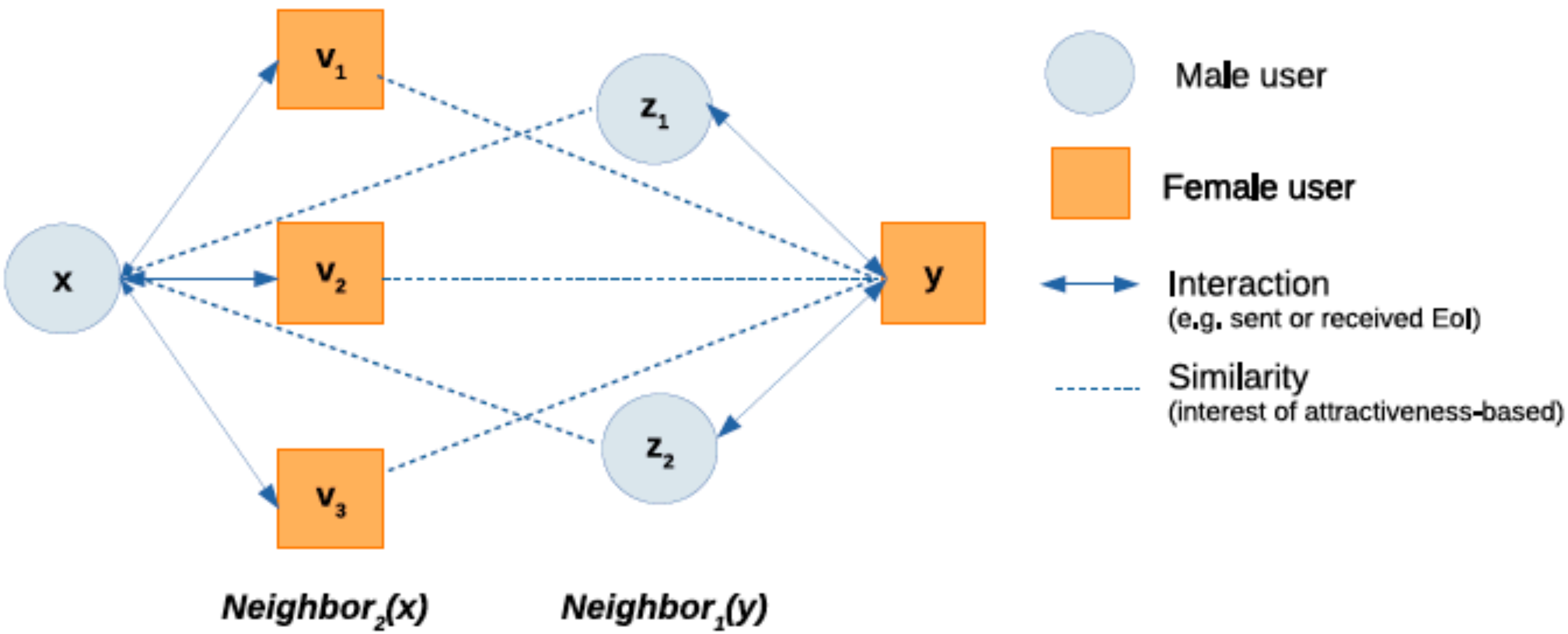
$$Neighbor(x) = EoI_{from}(x)$$

$$Neighbor(x) = EoI_{to}(x)$$

- Different combinations of **similarity measure + neighbourhood function** give rise to different "versions" of RCF.

# RCF: A collaborative filtering RRS

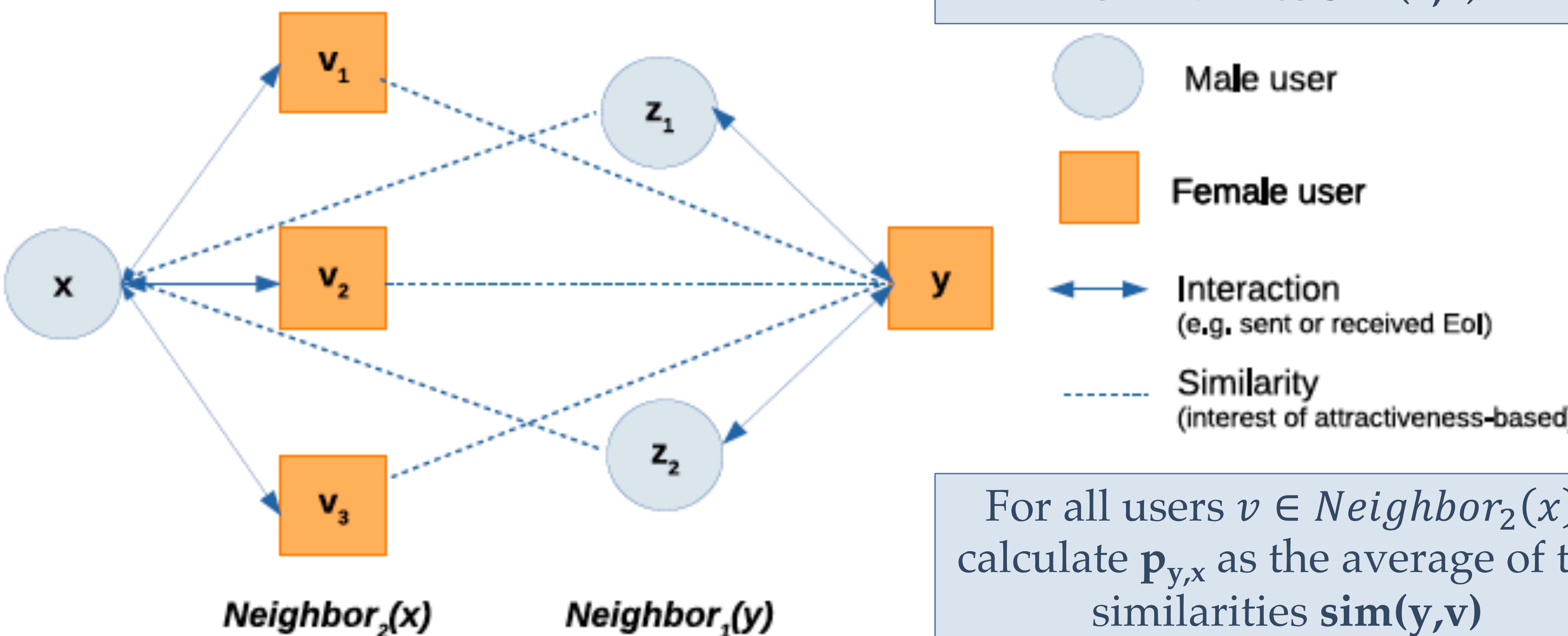
*General RCF procedure*



# RCF: A collaborative filtering RRS

*General RCF procedure*

For all users $z \in Neighbor_1(y)$, calculate $\mathbf{p_{x,y}}$ as the average of the similarities **sim(x,z)**




For all users $v \in Neighbor_2(x)$, calculate $\mathbf{p_{y,x}}$ as the average of the similarities **sim(y,v)**

# RCF: A collaborative filtering RRS

*General RCF procedure*

- **Preference fusion**: Reciprocal preference $p_{x \leftrightarrow y}$ calculated using the *harmonic mean*.
- Filter for $x$ the $N$ users $y$ with highest $p_{x \leftrightarrow y}$

**Results:**

- RCF more effective than prior CB-RRS in precision and recall.
- Relatively simple to implement and understand.
- Memory-based: no pre-trained model required.
- *Drawback*: Computational/temporal cost on large datasets.

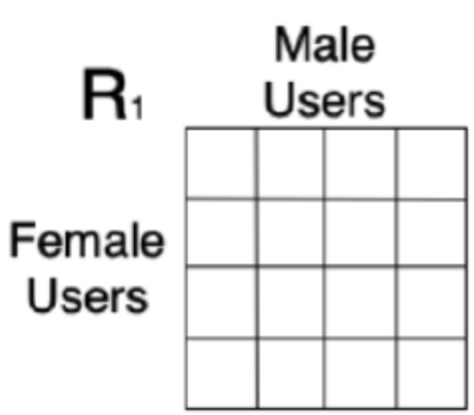
# LFRR: Latent Factor Model RRS

**LFRR** is a model-based CF-RRS that determines latent user attributes.

J. Neve, I. Palomares (2019). *Latent Factor Models and Aggregation Operators for Collaborative Filtering in Reciprocal Recommender Systems* (Long Paper). Proc. ACM Recsys'19, pp.219-227.

***Basic Ideas***

- **Two preference matrices**: *female-to-male preferences* and *male-to-female preferences*.
- Two LF models (one per matrix) trained via **matrix factorization**.



# LFRR: Latent Factor Model RRS

**LFRR** is a model-based CF-RRS that determines latent user attributes.

J. Neve, I. Palomares (2019). *Latent Factor Models and Aggregation Operators for Collaborative Filtering in Reciprocal Recommender Systems* (Long Paper). Proc. ACM Recsys'19, pp.219-227.

***Basic Ideas***

- **Two preference matrices**: *female-to-male preferences* and *male-to-female preferences*.
- Two LF models (one per matrix) trained via **matrix factorization**.

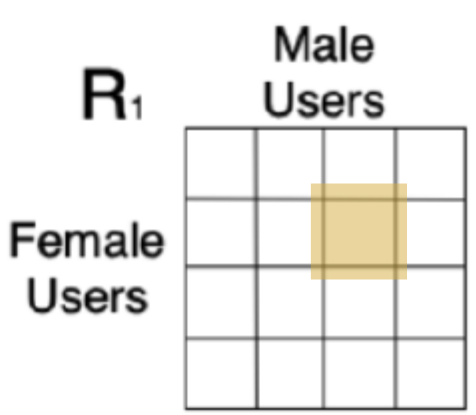
# LFRR: Latent Factor Model RRS

**LFRR** is a model-based CF-RRS that determines latent user attributes.

J. Neve, I. Palomares (2019). *Latent Factor Models and Aggregation Operators for Collaborative Filtering in Reciprocal Recommender Systems* (Long Paper). Proc. ACM Recsys'19, pp.219-227.

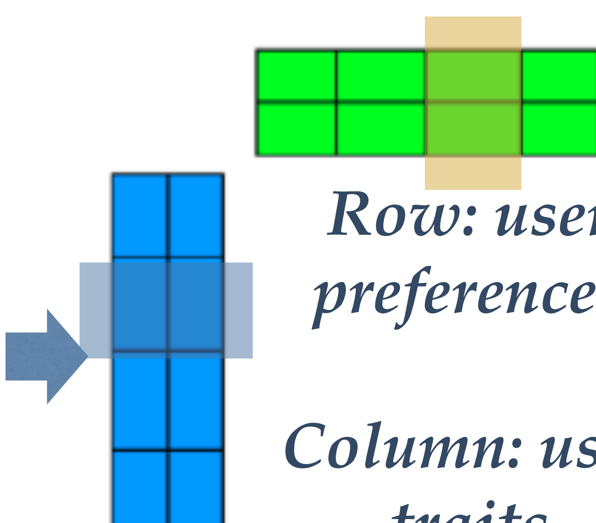
### *Basic Ideas*

- **Two preference matrices**: *female-to-male preferences* and *male-to-female preferences*.
- Two LF models (one per matrix) trained via **matrix factorization**.




***Stochastic Gradient Descent (SGD)***

$$\min_{q,p} \sum_{r_{x,y}\in R} (r_{x,y} - \mathbf{q}_y^T \mathbf{p}_x)^2 + \lambda(||\mathbf{q}_y||^2 + ||\mathbf{p}_x||^2)$$

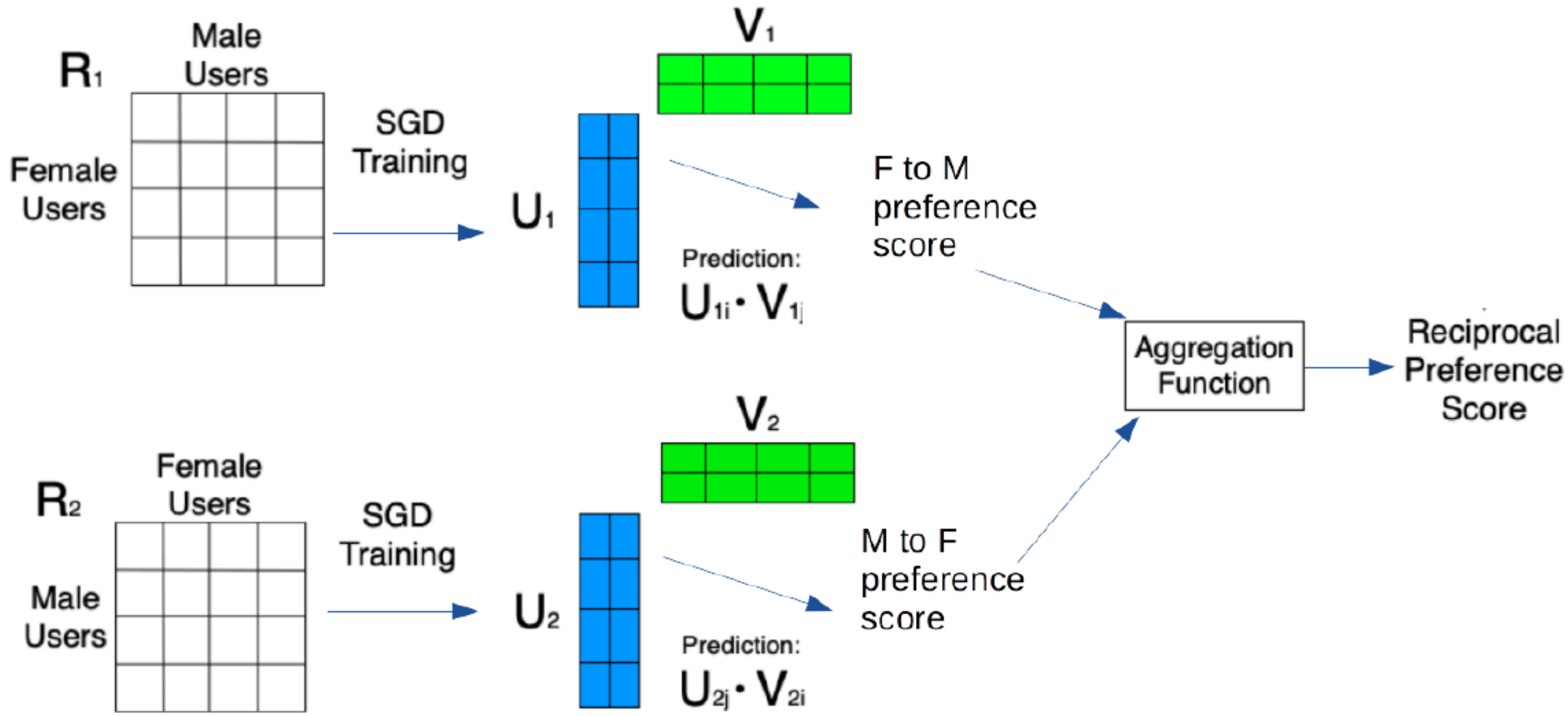


***Row: user preferences***

***Column: user traits***

# LFRR: Latent Factor Model RRS

Preference $p_{x,y}$ estimated as the resulting dot product of latent vectors, $\boldsymbol{p}_x \cdot \boldsymbol{q}_y^T$, after applying SGD.




$$\min_{q,p} \sum_{r_{x,y} \in R} (r_{x,y} - \mathbf{q}_y^T \mathbf{p}_x)^2 + \lambda(||\mathbf{q}_y||^2 + ||\mathbf{p}_x||^2)$$

# LFRR: Latent Factor Model RRS

**Results:**

- Tested against a large-scale real dataset with millions of users.

pairs

- Similarly promising performance to RCF (precision, recall and F1).
- Better efficiency: real-time recommendations under large datasets.

| Algorithm | Precision | Recall | Best F1 Score |
|---|---|---|---|
| *RCF (Arithmetic Mean)* | 0.58 | 0.86 | 0.69 |
| *RCF (Geometric Mean)* | 0.55 | 0.90 | 0.68 |
| *RCF (Harmonic Mean)* | 0.83 | 0.84 | 0.84 |
| *RCF (Uninorm)* | 0.84 | 0.90 | 0.87 |
| *LFRR (Arithmetic Mean)* | 0.81 | 0.92 | 0.87 |
| *LFRR (Geometric Mean)* | 0.83 | 0.86 | 0.85 |
| *LFRR (Harmonic Mean)* | 0.86 | 0.85 | 0.86 |
| *LFRR (Uninorm)* | 0.54 | 0.98 | 0.70 |

| Size | RCF Score | RCF List | LRFF Score | LFRR List |
|---|---|---|---|---|
| $10^3$ | 0.003 | 1.75 | $1 \times 10^{-}5$ | 0.0001 |
| $10^4$ | 0.005 | 13.7 | $1 \times 10^{-}5$ | 0.001 |
| $10^5$ | 0.008 | 163 | $1 \times 10^{-}5$ | 0.025 |
| $10^6$ | 0.09 | $>$ 1800 | $1 \times 10^{-}5$ | 0.63 |
| $10^7$ | 0.5 | $>$ 1800 | $1 \times 10^{-}5$ | 2.0 |

# Hybrid RRS

**Combine the strengths of more than one recommender technique**

- CB combined with CF.
- CB combined with knowledge-based recommendation.
- CF combined with other approaches, e.g. Hidden Markov Models (HMM) to predict next interaction based on *k* previous ones.
- Classical RS approaches combined with RRS (only when user-user and user-item interactions co-exist).

**Approaches**

- Sequential/cascade, e.g. CF followed by HMM (Alanazi & Bain, 2016).
- Community detection to address cold-start problem. (Yu et al., 2018).
- Incorporate facial features (Zhang et al., 2017).

# CCR: Content-Collaborative RRS

**CCR** is the first RRS model that integrates CB and CF.

J. Akehurst, et al. (2011). *CCR: A Content-Collaborative Reciprocal Recommender for Online Dating*. Proceedings IJCAI'11.

***Key Elements***

- Integrates distance metrics for CB and CF.
- **CB:** Distance between users based on profile attributes.
- **CF:** *"Similar people **[like / are liked by]** and **[dislike / are disliked by]** similar people".*
- Determine interaction groups for $x$ based on the two principles.

# CCR: Content-Collaborative RRS

***CCR Recommendation process***

- Take interaction data associated to $x$, e.g. $EoI_{from}(x)$, $EoI_{to}(x)$
- Define the following **interaction groups**:
  1. Users whom $x$ likes.
  2. Users who like $x$.
  3. Users whom $x$ dislikes.
  4. Users who dislike $x$.
  5. Users who *reciprocally* like $x$ (intersection of 1 and 2).

- **Apply CB first**: determine set of users $S_x$ with similar profile to $x$.

# CCR: Content-Collaborative RRS

*CCR Recommendation process*

- Once we have $S_x$, apply **CF by analysing user interactions**:
  - For every $z \in S_x$, determine list of users (s)he had reciprocal interest with.
  - This produces several lists of candidates $v$, one list for each $z$.
  - Calculate each candidate's support to finally rank them for $x$.

$$\mathbf{Sup}(v, S_x) = \#\,\textbf{positive interactions} - \#\,\textbf{negative interactions}$$

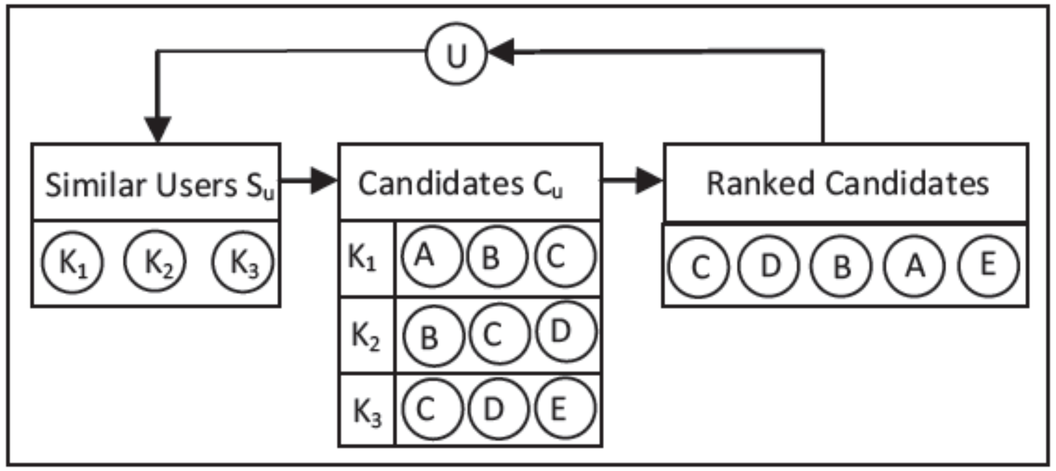



Image source: Akehurst et al. (2011)

# CCR: Content-Collaborative RRS

***Results***

- Evaluation against a baseline approach picking $S_x$ at random.
- Success rate almost doubled (70%): % recommended users who sent/received an EoI to/from $x$ and response was positive.
- Cold-start problem alleviated.
- Memory-based → likely to be less efficient against large data.

Image source: Akehurst et al. (2011)

# HRRS: A Hybrid classical-RRS

**HRRS** combines principles from classical user-item recommendation in the reciprocal recommendation process.

- Designed to connect users in platforms/apps where they publish and share ***content*** → skill-sharing apps, e.g. *Cookpad*.

J. Neve, I. Palomares. (2020). *Hybrid Reciprocal Recommender Systems: integrating item-to-user principles in reciprocal recommendation*. Companion Proc. ACM Webconf'20, pp. 848-854

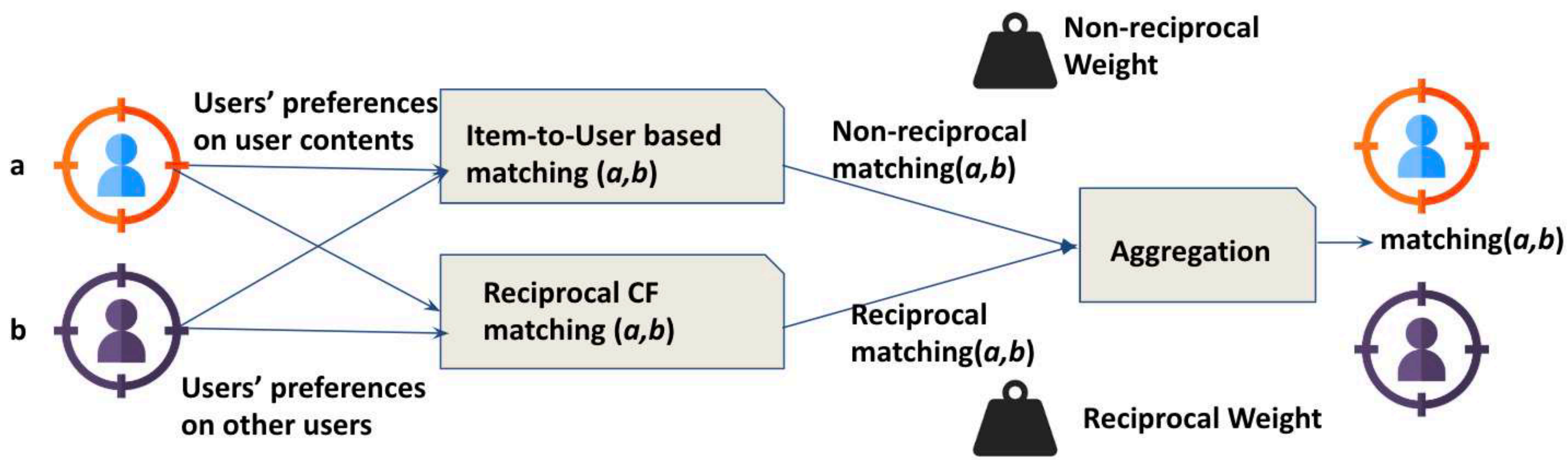
***Key Elements***

- User-to-user preference indicators and user-to-content preference indicators coexist.
- Single-class RRS, no distinction between different classes of users.

# HRRS: A Hybrid classical-RRS

*HRRS model*

i. **Item-to-user matching**: Based on classical CF → users' similarity on content liked. User similarities on liked content is inherently *symmetric*.
ii. **Reciprocal matching**: Based on users' preferences to other users (requires reciprocity)
iii. **Aggregation**: Combination of both approaches into a final matching between two users.



# HRRS: A Hybrid classical-RRS

***HRRS model: Item-to-user Matching***

- Pairwise user similarities based on preferences towards content published by any other users: bookmarked recipes.
- Unary ratings: preference by user *x* towards content *c* is either "like" or "unknown" → **Jaccard Index** $\frac{R_a \cap R_b}{R_a \cup R_b}$

- Problematic in domains with many contents items that are very similar to each other.
  - Example: user *a* liked "*potato omelette*" recipe, user *b* liked "*Spanish potato omelette*".
  - Jaccard Index only accounts for co-occurrences of exactly the same item.

# HRRS: A Hybrid classical-RRS

*HRRS model: Item-to-user Matching*

- **Solution**: "Soft" extension of Jaccard index that captures similar but not identical items that two users might have liked.
- Adjustment terms l,μ introduce non-zero similarities between non identical items.
- For quantifying text information (e.g. recipe titles and ingredients) we use *Word2Vec*, producing **word embeddings** associated to pairs of content items $r_a$, $r_b$

$$\frac{|R_a \cap R_b| + \lambda}{|R_a \cup R_b| + \mu} \Leftarrow \lambda = \sum_{r_a \in R_a - R_b} \sum_{r_b \in R_b - R_a} \delta(r_a, r_b) \quad \mu = |R_a - R_b| \cdot |R_b - R_a| \Leftarrow \delta(r_a, r_b) = \sum_{l=1}^{|r_a|} \sum_{k=l+1}^{|r_b|} sum(w_l, w_k)$$

# HRRS: A Hybrid classical-RRS

*HRRS model: Reciprocal Matching*

- **Preference indicators**: Follows to other users.
- Single-class Latent Factor Model (based on LFRR model).
- User $x$ preference to user $y$ properties.
- User $y$ preference to user $x$ properties.
- Harmonic mean to aggregate unidirectional user preferences into reciprocal score.

*Aggregation of user-user matchings*

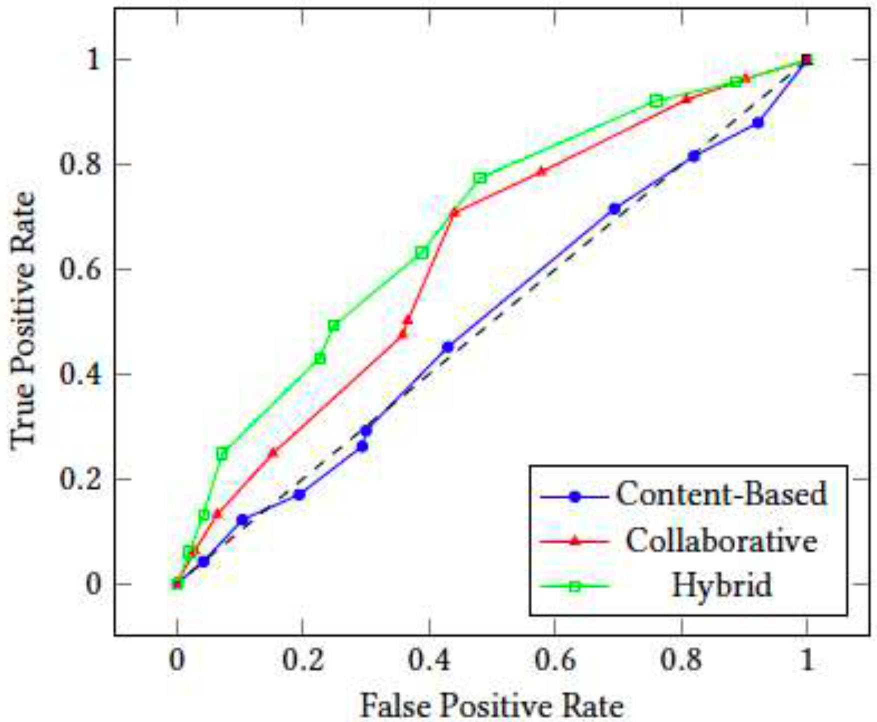
- The item-to-user and reciprocal matching scores $m(x,y)$ are aggregated using a simple weighted mean.

# HRRS: A Hybrid classical-RRS

*Results*

- Offline evaluation with 500 user pairs from *Cookpad* website, 250 of which indicated mutual preference through *Follows*.
- Predict a positive match (+) if the *m(x,y)* value returned by HRRS is above a given threshold, and a negative match (-) otherwise.
- **Two baselines**: reciprocal only; content preference only.

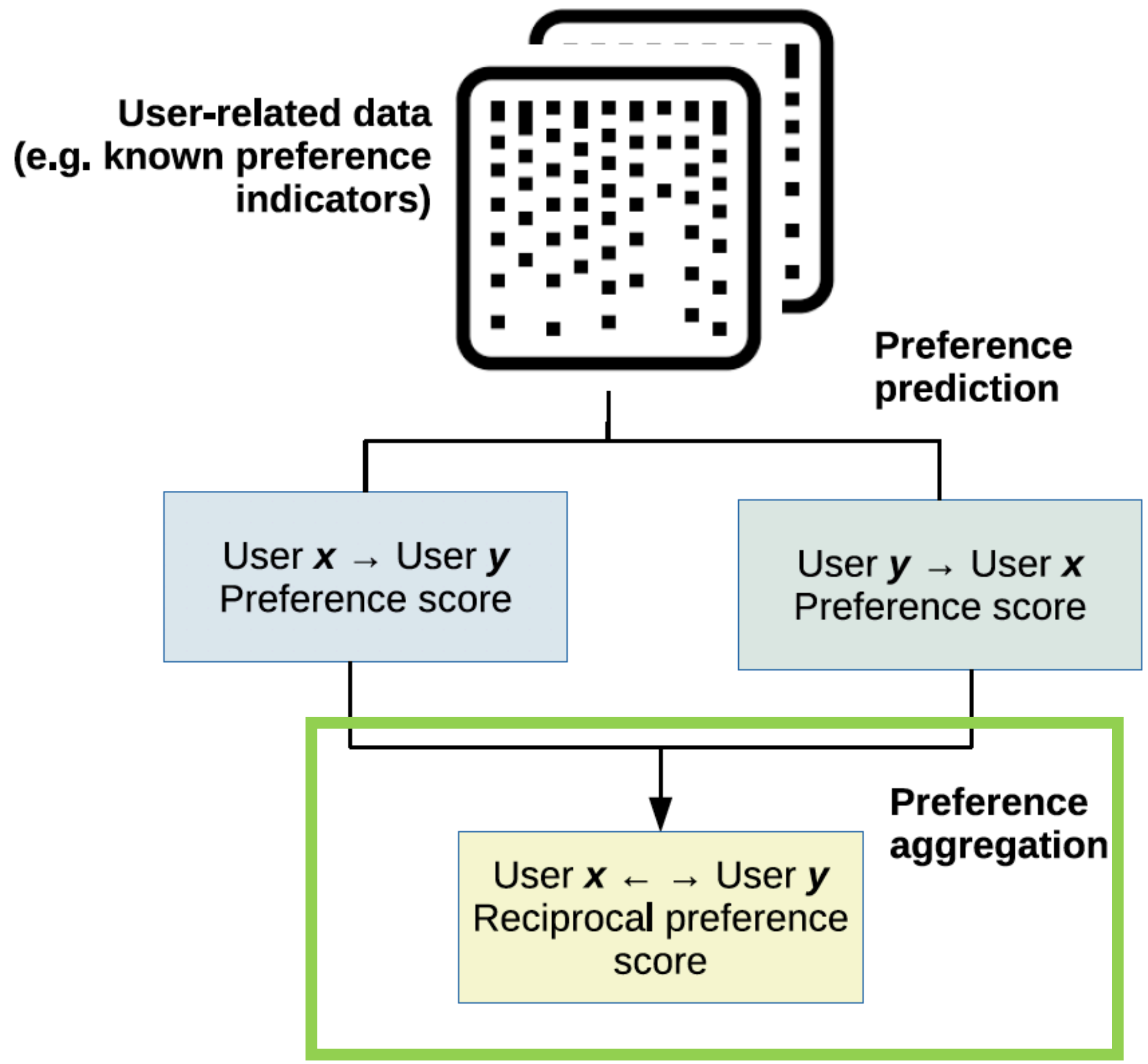
| Threshold | Rec. F1 Score | NonRec. F1 Score | HRRS F1 Score |
|---|---|---|---|
| 0 | **0.666** | **0.666** | **0.666** |
| 0.1 | 0.672 | 0.628 | **0.673** |
| 0.2 | 0.676 | 0.619 | **0.687** |
| 0.3 | 0.664 | 0.594 | **0.686** |
| 0.4 | **0.659** | 0.480 | 0.625 |
| 0.5 | 0.537 | 0.366 | **0.565** |
| 0.6 | 0.517 | 0.336 | **0.519** |
| 0.7 | 0.354 | 0.249 | **0.375** |
| 0.8 | 0.220 | 0.199 | **0.224** |
| 0.9 | **0.113** | 0.077 | 0.111 |
| 1.0 | 0 | 0 | 0 |



# The reciprocal recommendation process II: Fusion

# Fusion process: measuring *reciprocity*

## General RRS conceptual model



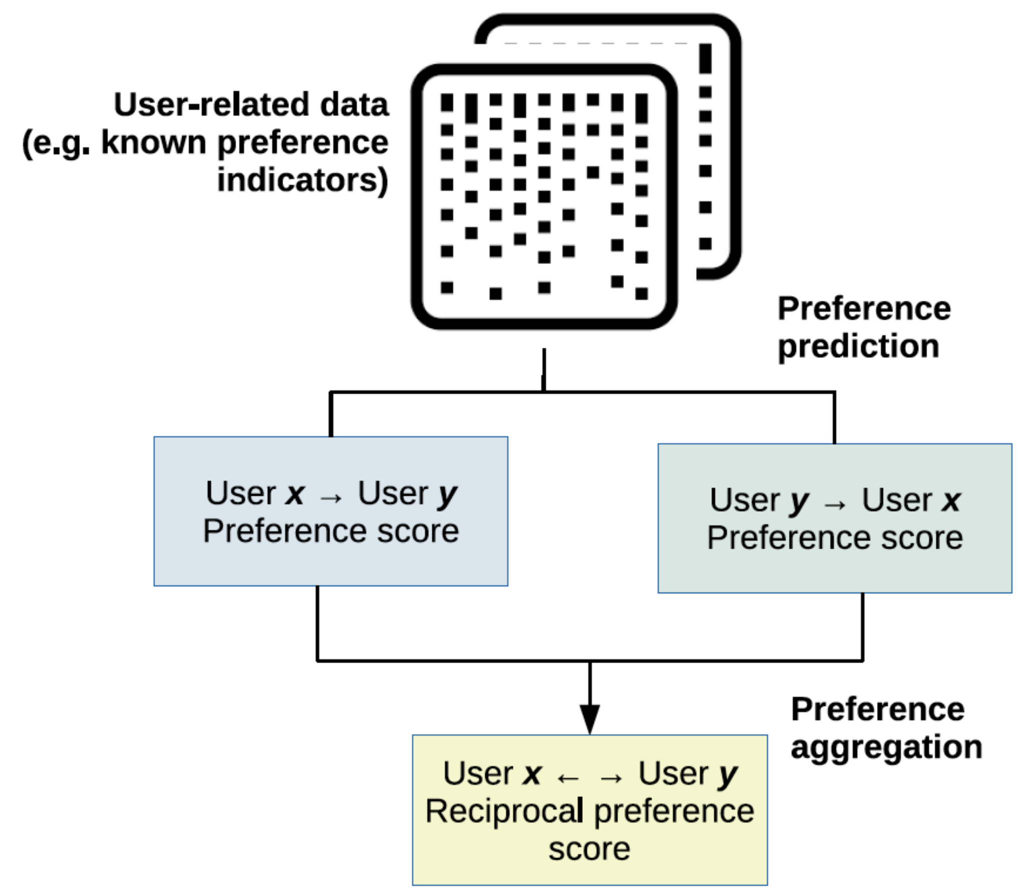

*We are HERE!*

# Fusion process: measuring *reciprocity*

**Reciprocity** is arguably the most fundamental and differentiating complexity in RRS, with respect to other RS.

The most commonly followed approach to account for reciprocity consists in calculating an aggregated mutual preference score…

… but it is not the only *fusion* strategy to capture reciprocity. Let's have a look at some of them.



# Fusion process: measuring *reciprocity*

| **Fusion methods for capturing reciprocity in RRS literature** |
|---|
| Aggregation of numerical preference scores $p_{x,y}$ and $p_{y,x}$<br>Harmonic mean, other means, sum, product, weighted mean, mixed aggregation… |
| Matrix multiplication |
| Set intersection of recommendation lists |
| Aggregation of probabilities |
| Conjuction (AND-like) |
| Community-level matching |
| Rank aggregation on recommendation lists |

# Fusion process: aggregation of preferences

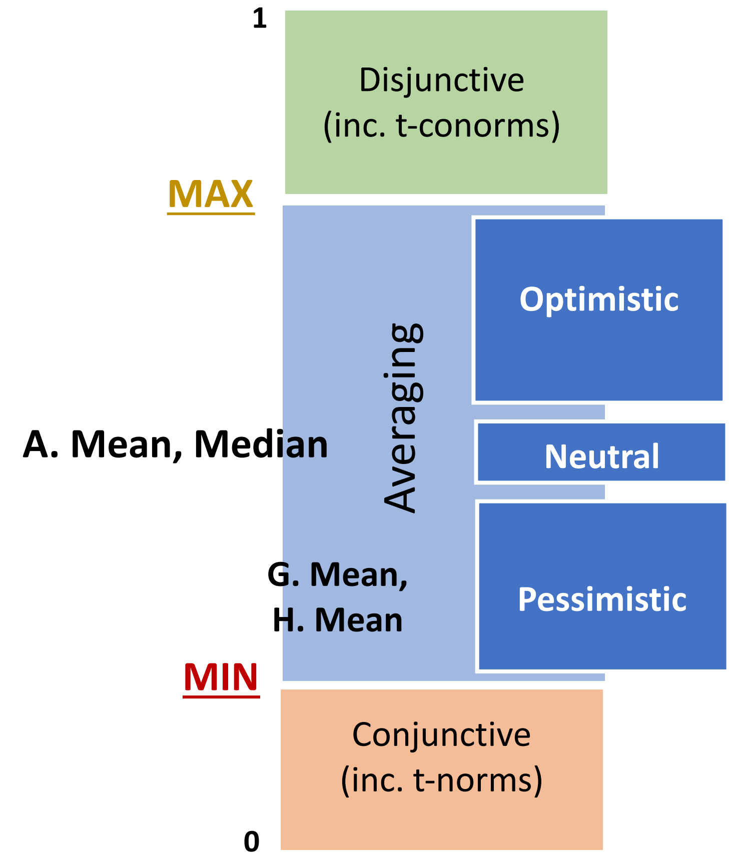
***Definition of Aggregation Function***

$\boldsymbol{a(x) = f(a_1(x), \ldots, a_n(x))}$, with f:$[0,1]^n \rightarrow [0,1]$

Basic Properties

f(0,…0) = 0 and f(1,…,1)=1 *(Boundary)*

$\boldsymbol{x \leq y}$ implies f($\boldsymbol{x}$) ≤ $f(\boldsymbol{y})$ with $\boldsymbol{x = [x_1 \ldots x_n]}$ and $\boldsymbol{y = [y_1 \ldots y_n]}$ *(Monotonicity)*




In RRS, we are normally interested in the particular case of two inputs:

**f:$[0,1]^2 \rightarrow [0,1] = [0,1]\text{x}[0,1] \rightarrow [0,1]$**

# Fusion process: aggregation of preferences

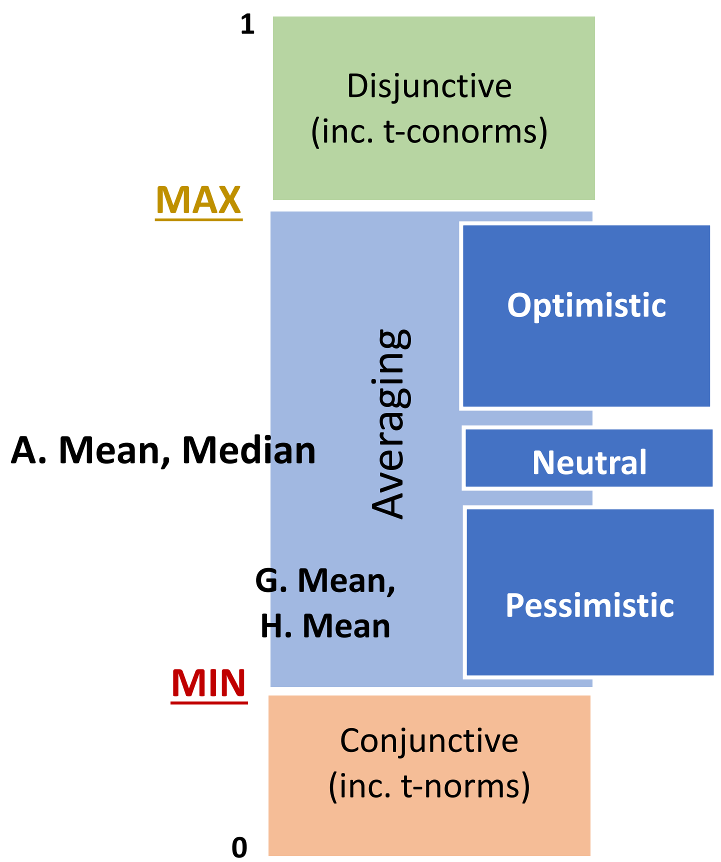
**Weighted vs Non-weighted**

- Importance weights assigned to experts/criteria

**Attitudinal Character of Aggregation**

- Disjunctive vs Averaging vs Conjunctive (vs Mixed)
- Optimistic vs Neutral vs Pessimistic

Choosing the right aggregation function is a central aspect in any RS for combining preferences, particularly in **Group RS**, **multicriteria RS** and **RRS**.



# Fusion process: aggregation of preferences

**Arithmetic mean (AM), Geometric mean (GM), Harmonic Mean (HM)**

- AM strictly higher than GM, and GM strictly higher than HM, for non-negative and non-identical inputs.
- In situations when inputs equal or close to 0, GM or HM may severely affect the output (e.g. average mark of a student).
- GM popular in business and finance, e.g. to deal with percentages, calculate growth rates, financial stock market indices, …
- **HM is more stable to outliers,** when there are very high (resp. low) inputs. Useful when all inputs should be reasonably high to yield a reasonably high output → matching people in RRS applications!

$$A = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n}$$

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} = \left(\frac{\sum_{i=1}^{n} x_i^{-1}}{n}\right)^{-1}.$$

# Fusion process: aggregation of preferences

**Weighted means**

The **Weighted Arithmetic Mean** allows for assigning importance degrees to each of the elements to aggregate. Elements with highest weights will be more influential in the aggregated output value.

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i},$$

If used in an **RRS**, how to adequately weigh the two parties' unidirectional preferences? Based on what? ?

The GM and HM also have their weighted counterparts:

$$\bar{x} = \left(\prod_{i=1}^{n} x_i^{w_i}\right)^{1/\sum_{i=1}^{n} w_i} = \exp\left(\frac{\sum_{i=1}^{n} w_i \ln x_i}{\sum_{i=1}^{n} w_i}\right)$$

$$H = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}} = \left(\frac{\sum_{i=1}^{n} w_i x_i^{-1}}{\sum_{i=1}^{n} w_i}\right)^{-1}.$$

# Fusion process: aggregation of preferences

<u>**Conjunctive and disjunctive aggregation**</u>

***T-norms are generalizations of the logical conjunction → conjunctive aggregation functions (aggregation result below the minimum)***

Examples:

-<u>Minimum</u>: T(a,b) = min(a,b)

-<u>Product</u>: T(a,b) = a*b

-<u>Lukasiewicz t-norm</u>: T(a,b) = max{0, a+b-1}

***T-Conorms are generalizations of the logical disjunction → disjunctive aggregation functions (aggregation result above the maximum)***

Examples:

-<u>Maximum</u>: S(a,b) = max(a,b)

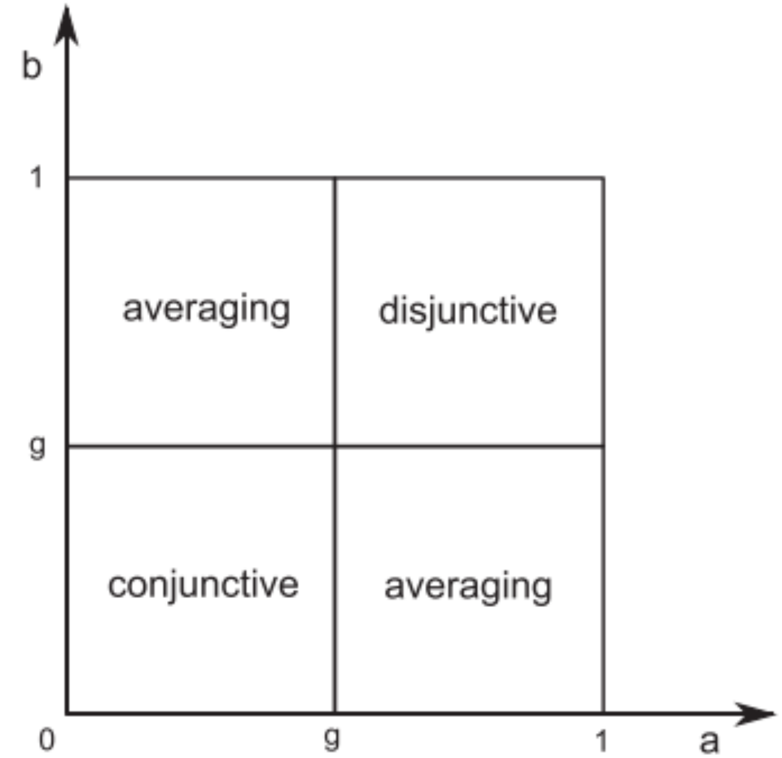
-<u>Probabilistic sum</u>: S(a,b) = a + b - a*b

-<u>Bounded sum</u>: S(a,b) = min{a+b, 1}

# Fusion process: aggregation of preferences

**Mixed-behaviour aggregation: uninorm functions**

- A **uninorm** is a function U: $[0, 1]^2 \rightarrow [0, 1]$ which is commutative, associative, monotone, and has a **neutral element** ***g* ∈ *[0,1]***
- It fulfils the **FULL REINFORCEMENT PROPERTY:** two high (resp. low) values reinforce each other, giving a higher (resp. lower) aggregated output.




R.R. Yager. *Uninorm Aggregation Operators*, Fuzzy Sets and Systems, 80(1), pp. 111-120, 1996.

# Fusion process: aggregation of preferences

## Mixed-behaviour aggregation: uninorm functions

**Example** The cross-ratio uninorm is a continuous uninorm in $[0, 1]^2 \setminus \{(0, 1), (1, 0)\}$, with neutral element $g = 0.5$:

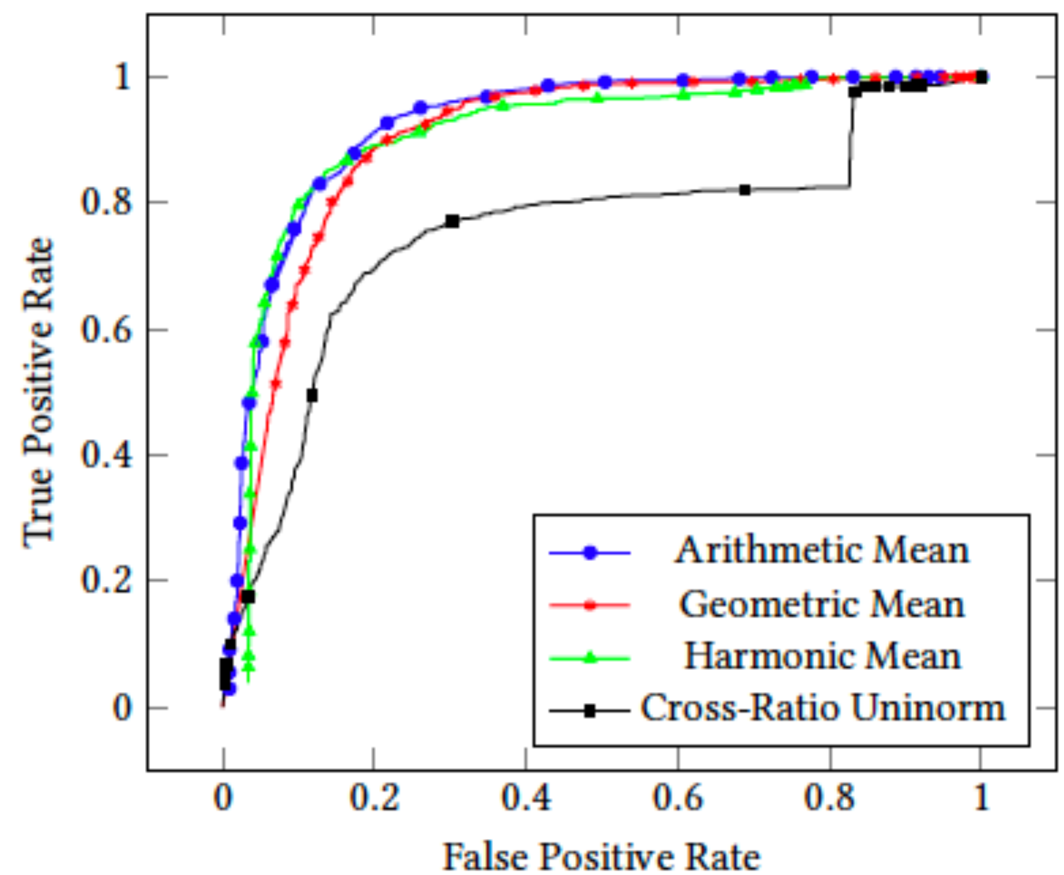
$$\mathcal{U}(a, b) = \begin{cases} 0 & \text{if } (a, b) \in \{(0, 1), (1, 0)\}, \\ \dfrac{ab}{ab + (1 - a)(1 - b)} & \text{otherwise.} \end{cases}$$

| $p_{x,y}$ | $p_{y,x}$ | $p_{x \leftrightarrow y}$ |
|---|---|---|
| 0.612 | 0.409 | **0.532** |
| 0.38 | 0.415 | **0.303** |
| 0.5 | 0.437 | **0.437** |
| 0.675 | 0.58 | **0.741** |

# Fusion process: aggregation of preferences

**Putting a few aggregation functions together into an RRS**

- Experiments with **LFRR model**.
- Dataset from Japanese dating site.
- Effect of using different aggregation operators than the *harmonic mean* to fuse pairs of users' preferences.




***Results***

- Arithmetic mean outperforms in *recall*.
- Cross-ratio uninorm doesn't seem very promising.

| Algorithm | Precision | Recall | Best F1 Score |
|---|---|---|---|
| *LFRR (Arithmetic Mean)* | 0.81 | 0.92 | 0.87 |
| *LFRR (Geometric Mean)* | 0.83 | 0.86 | 0.85 |
| *LFRR (Harmonic Mean)* | 0.86 | 0.85 | 0.86 |
| *LFRR (Uninorm)* | 0.54 | 0.98 | 0.70 |

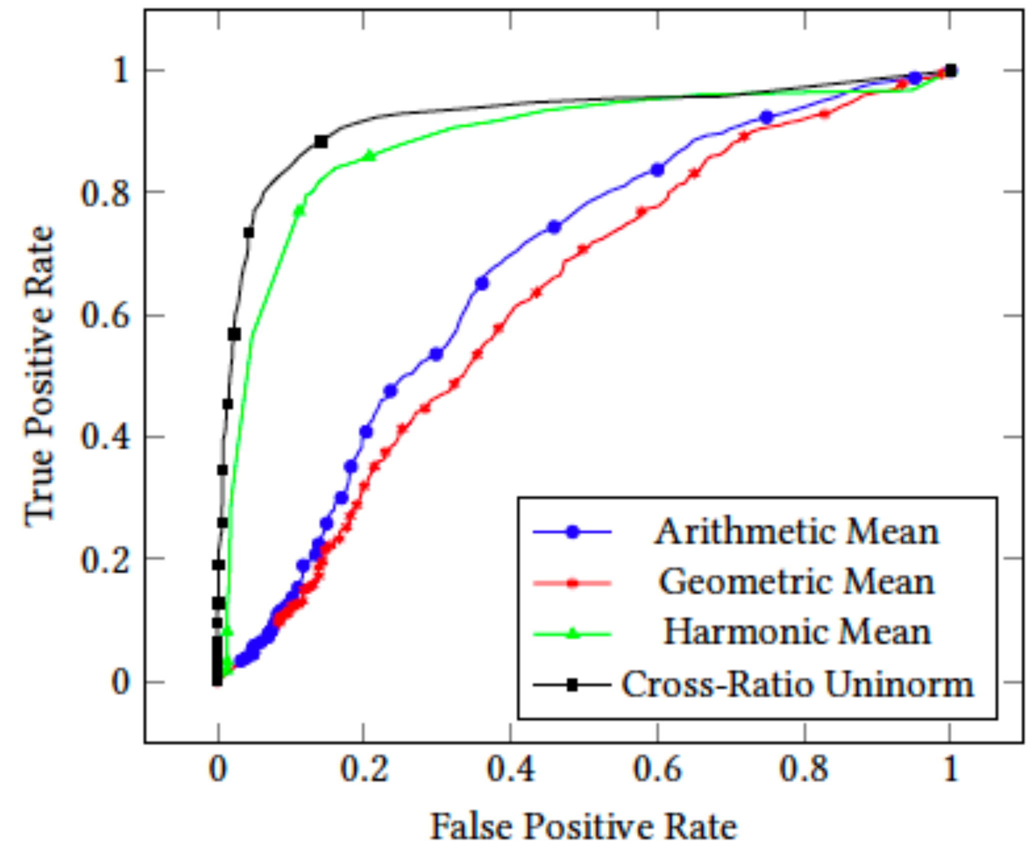
# Fusion process: aggregation of preferences

**Putting a few aggregation functions together into an RRS**

- Experiments with **RCF model**.
- Dataset from Japanese dating site.
- Effect of using different aggregation operators than the *harmonic mean* to fuse pairs of users' preferences.

***Results***

- Cross-ratio outperforms *HM* in precision and F1 score!




| Algorithm | Precision | Recall | Best F1 Score |
|---|---|---|---|
| *RCF (Arithmetic Mean)* | 0.58 | 0.86 | 0.69 |
| *RCF (Geometric Mean)* | 0.55 | 0.90 | 0.68 |
| *RCF (Harmonic Mean)* | 0.83 | 0.84 | 0.84 |
| *RCF (Uninorm)* | 0.84 | 0.90 | 0.87 |

J. Neve, I. Palomares. (2019). *Aggregation Strategies in User-to-User Reciprocal Recommender Systems*. Proc. IEEE SMC'19, pp. 4031-4036.

# Fusion process: aggregation of preferences

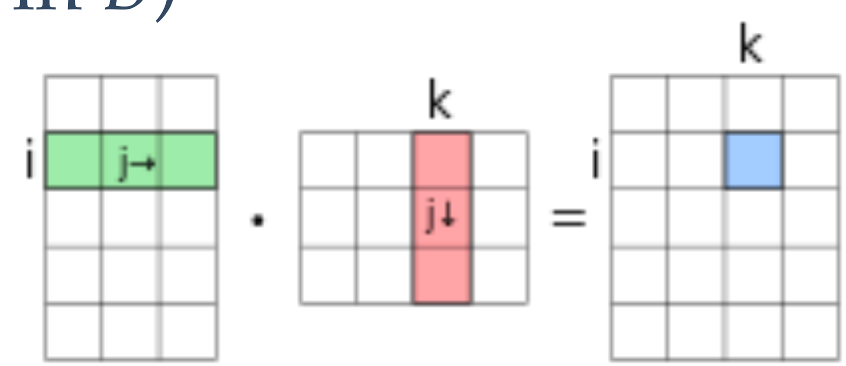
**Sky is the limit…**

Choquet Integral
Dempster Combination Rule
Conjunctive
Sugeno Integral
Averaging
Disjunctive
Weighted Mean
Power Means
Geometric Means
OWA
T-norm
Uninorm
T-conorm
Bonferroni Means

# Fusion process: matrix multiplication

- Used by (Jacobsen and Spanakis, 2019) in a system for matching graduate students and jobs.
- Define a **student-course** matrix $A$, where element $grade_{s,c}$ is the grade obtained by student $s$ in course $c$.
- Define a **job-course** matrix $B$, where element $sim_{j,c}$ describes the similarity between the characteristics of job $j$ and course $c$ contents.
- Each element of the product $AB^T$ is the matching between student $s$ (row vector in $A$) and job $j$ (column vector in $B$)

$$score_{s,j} = \sum_{c=1}^{M} grade_{s,c} * sim_{j,c}$$



# Fusion process: weighted mean

- Used by (Kleinerman et al., 2018), where the mutual preference score is calculated as:

$$p_{x \leftrightarrow y} = \alpha_x \cdot p_{x,y} + (1 - \alpha_x) p_{y,x}$$

- The weighting parameter $\alpha_x \in [0,1]$ is optimized for each subject user $x$, namely for optimizing successful interactions.
- In other words, **finding an optimal balance between subject user and object user** that maximizes successful interactions in the recommendations received.

# Fusion process: recommendation rankings

- Used by (Mine et al., 2013), in a job matching RRS.
- Let $s_i$ be a job seeker and $r_j$ a recruiter.
- Two *unidirectional* recommendation lists are created, one for the job seeker and one for the recruiter.
- The positions assigned to $s_i$ in the recommendation list for $r_j$, and vice versa, are aggregated to yield the matching score:

$$MatS(s_j, r_j) = \frac{1}{rank(s_i(r_j)) \cdot rank(r_j(s_i))}$$

| $rank(s_i(r_j))$ | $rank(r_j(s_i))$ | $matching$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 2 | 0.5 |
| 1 | 5 | 0.2 |
| 2 | 3 | 0.167 |
| 3 | 4 | 0.083 |

# Applications of RRS

# RRS Applications

**(1) Online Dating**

- Most popular target domain for RRS innovations.
- Earliest on RS for online dating, from around 2007 → Nonreciprocal
- First reciprocal approaches, primarily CB (2010-2013).
- Gentle shift towards CF and hybrid afterwards.

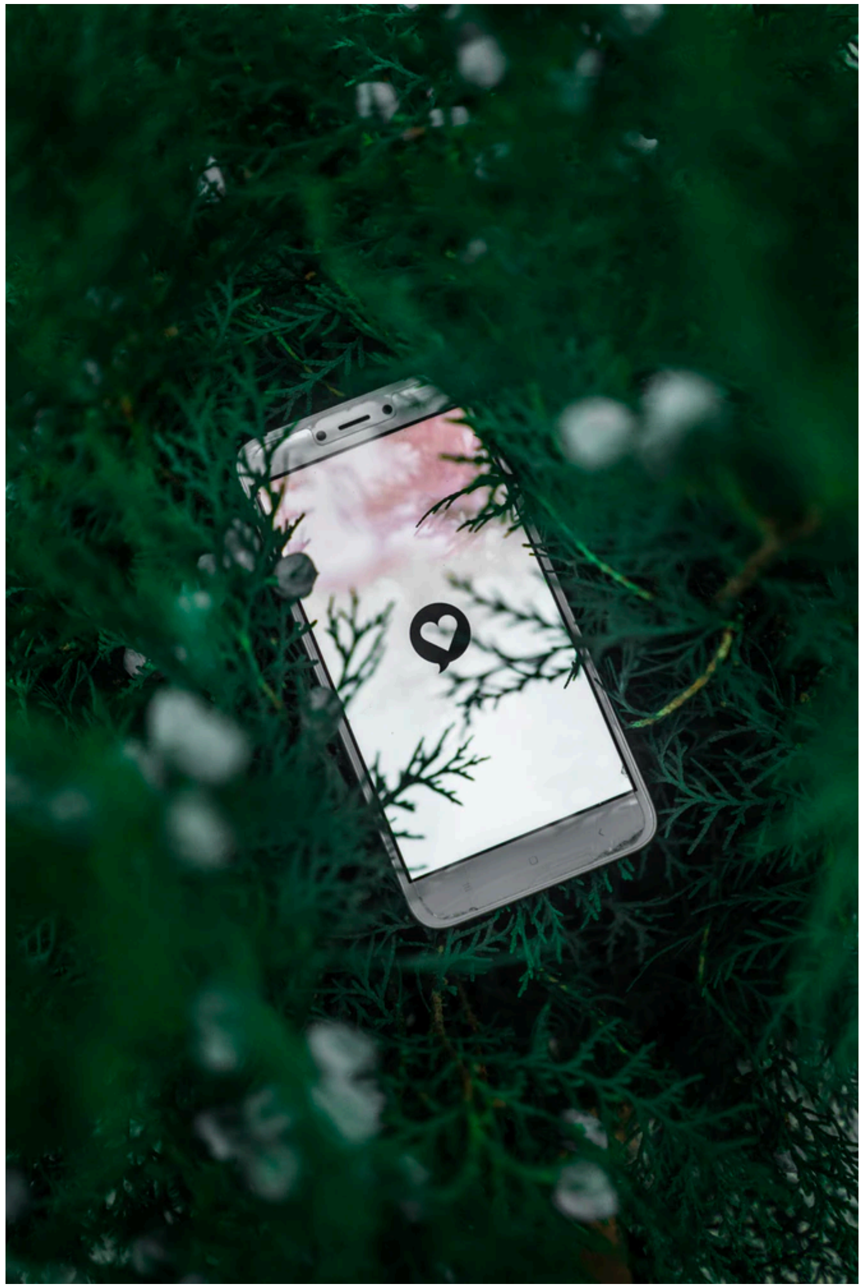


Image source: @graylab

# RRS Applications

## (1) Online Dating – theoretical studies

- Comprehensive case studies upon several success and evaluation metrics.
- Sensitivity to detect scammers.
- Temporal behavior, messaging and replying patterns of users.
- User correlations under different attributes.
- Deviation between explicit preferences and implicit preferences inferred from behavior.

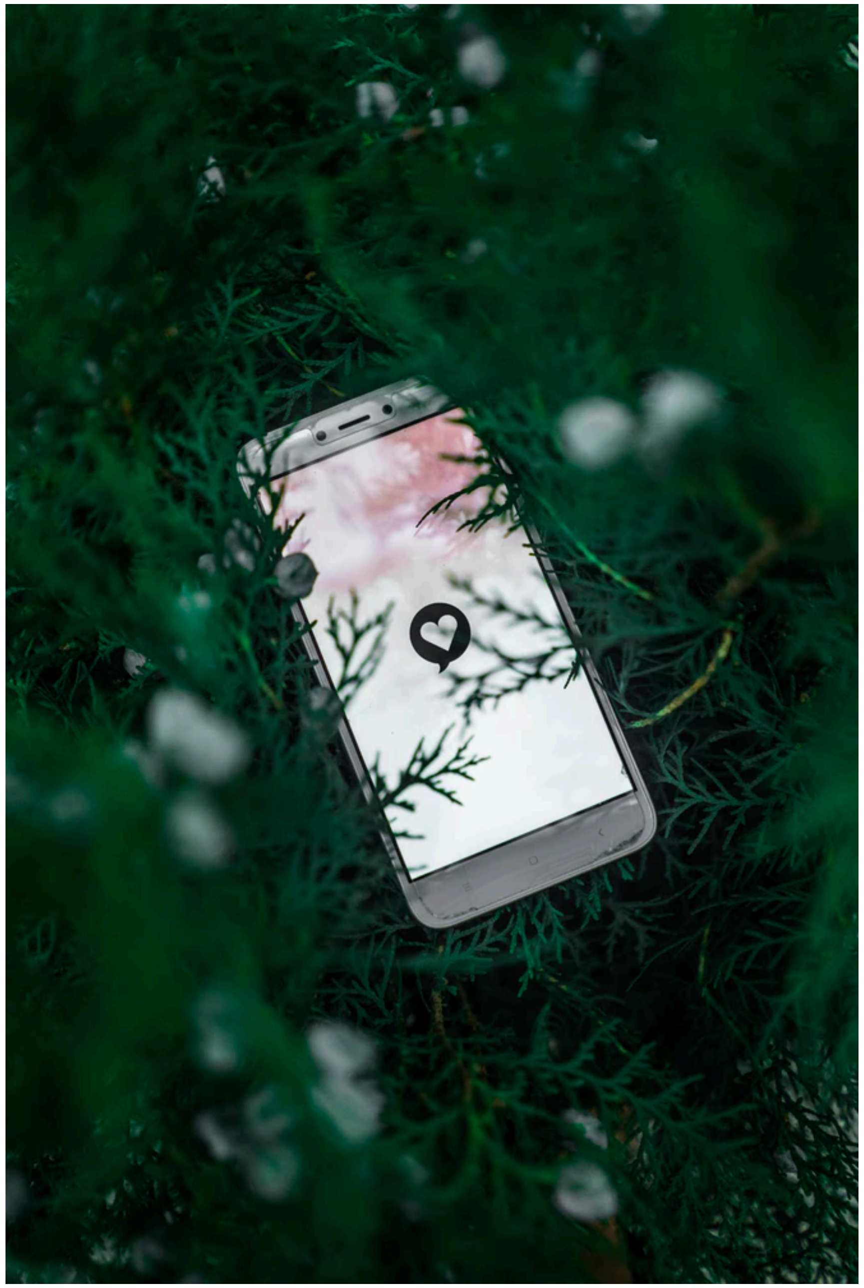


Image source: @graylab

# RRS Applications

**(1) Online Dating – theoretical studies**

- Finding "invisible communities" from messaging graphs, with clusters defined by *homophily* and *attractiveness*.
- Potential of facial attractiveness information mined from user pictures to overcome sparsity.
- Gender attribute differences in decision-making to choose a date.
- Computational complexity analyses.

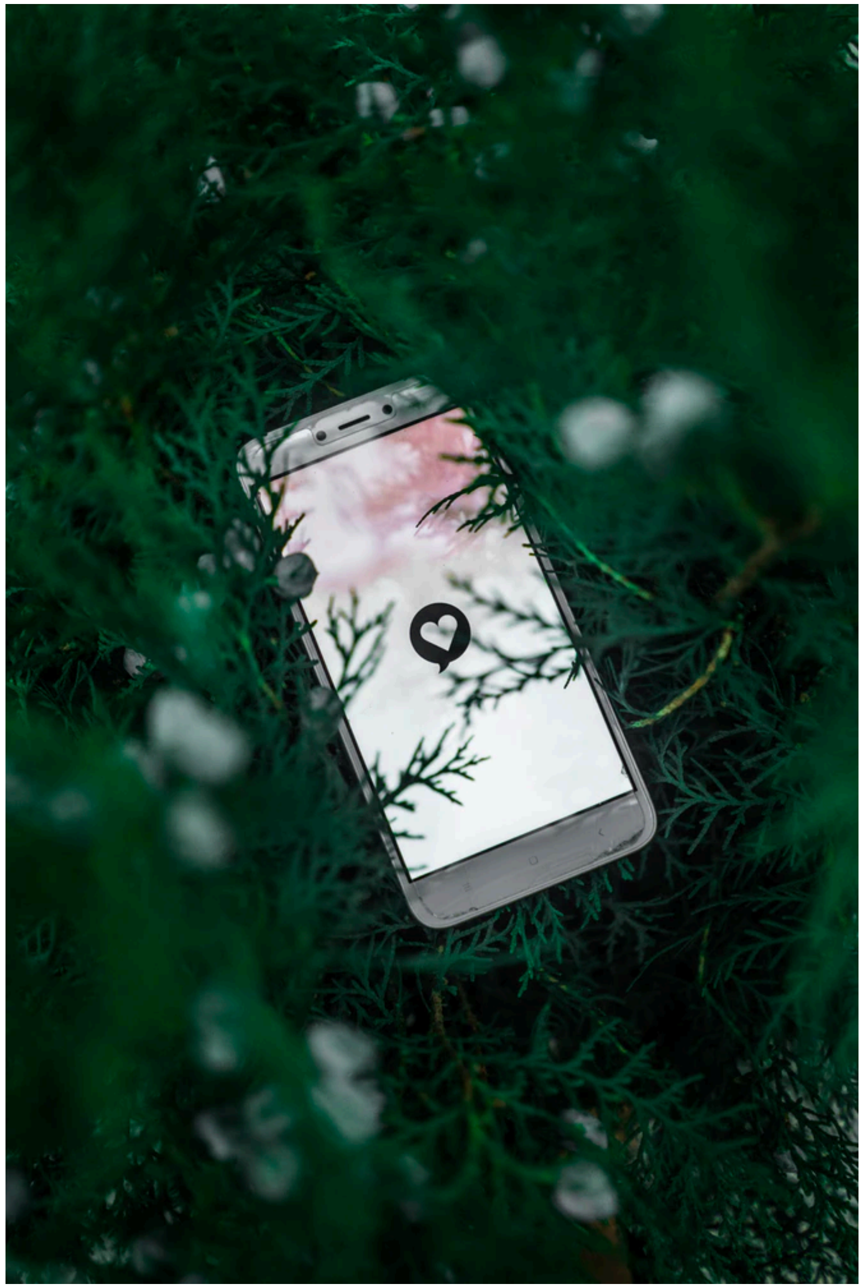


Image source: @graylab

# RRS Applications

**(1) Online Dating – models/techniques**

- Preferences as frequency distributions of attributes' values (RECON).
- Hidden Markov Models to predict next interaction(s) based on past ones.
- Explicit preferences from questionnaires.
- Deep learning on social media text.
- Bipartite network (RCF).
- Clustering similar users.
- Tensor models of user attributes-interactions.
- …

Image source: @graylab

# RRS Applications

**(2) Recruitment**

- Scenarios where a job seeker looks for recommendations, and both her/his interests and the recruiter interests need to align.
- Less RRS than in online dating, but good diversity of strategies.
- A few works focused on graduate students' recruitment.

Image source: @huntersrace

# RRS Applications

**(2) Recruitment – theoretical studies**

- Characterization of online recruitment services and monitoring demand-offer of employment.
- Correlations between explicit and implicit job preferences.
- Identifying best indicators of job seekers' preferences (implicit).
- Impact of reciprocity *versus* nonreciprocity.

Image source: @huntersrace

# RRS Applications

**(2) Recruitment – models/techniques**

- Analyzing past successful graduates for reciprocal graduate-job recommendation.
- Binary classification to predict clicks on jobs.
- Privacy oriented stable matching.
- Walrasian Equilibrium multi-objective optimization → fairness.
- Deep matrix factorization.

Image source: @huntersrace

# RRS Applications

**(3) Online Learning**

- Often large and diverse body of learners.
- Increasingly popular (or needed…).
- Predominantly CB approaches based on exploiting learners and teachers' profile information.
- Variety of scenarios:
  - Peer matching in MOOCs and university courses.
  - Group formation.
  - Learner-question matching in forums.
  - Student-supervisor matching.

Image source: @andyfalconerphotography

# RRS Applications

**(3) Online Learning – theoretical studies**

- Effects of peer recommendation in learner engagement.
- Study of recommendation strategies in MOOCs.

***Models***

- Peer matching based on RECON (attribute frequency distributions).
- Peer matching based on compatibility criteria.
- Group formation via optimization from individual recommendation lists.
- CB-CF for student-supervisor allocation.

Image source: @andyfalconerphotography

# RRS Applications

**(4) Online Social Networks**

- Social network sites can be symmetric or asymmetric.
- Symmetric SNSs require both sides to confirm the relationship (e.g. Facebook, LinkedIn) → **RRS**
- Asymmetric SNSs: one user can follow or connect to another user, without a requirement the other way round.
- Asymmetric SNS tend to show much more skewed in-degree and out-degree distributions.

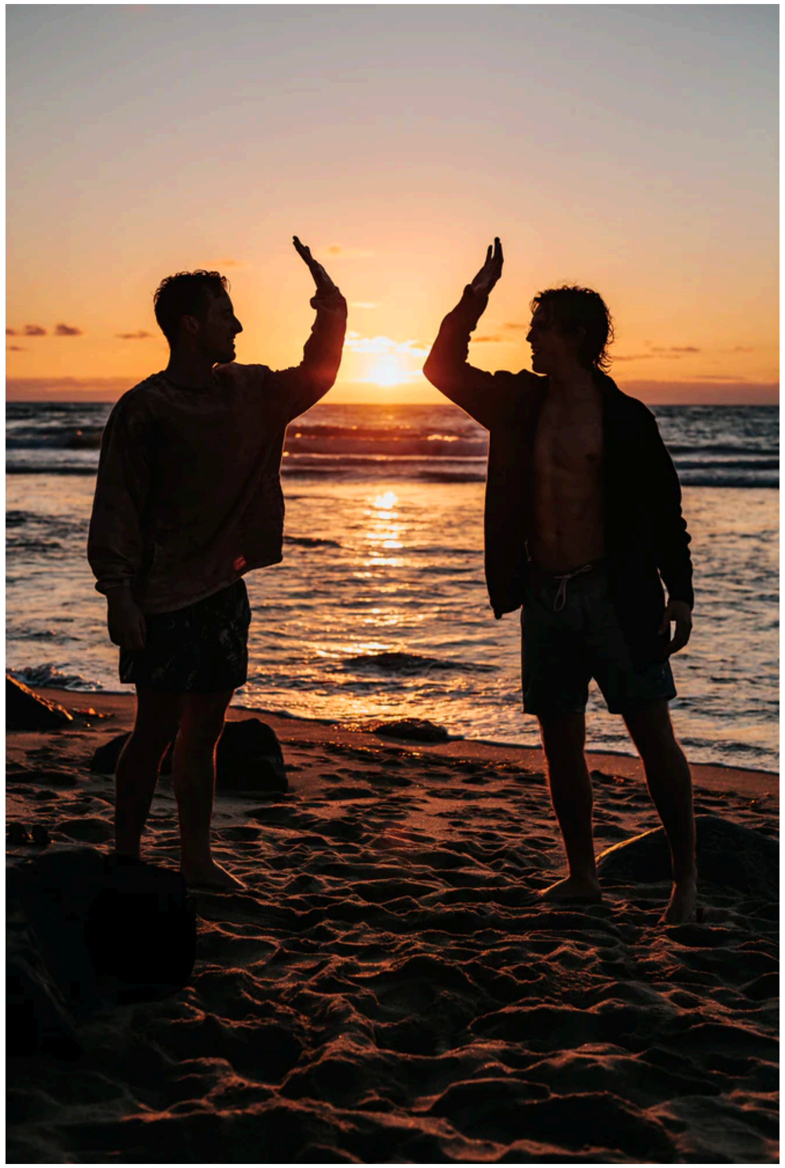


Image source: @jtylernix

# RRS Applications

**(4) SNSs – theoretical studies and models**

- Explanations in the success of people recommendation.
- Recommendation of unfamiliar people (strangers).
- Network structure characteristics (distribution, centrality, etc.).
- Trust and reputation-based recommendation.
- Proximity-driven link prediction.
- Genetic algorithms on graphs.

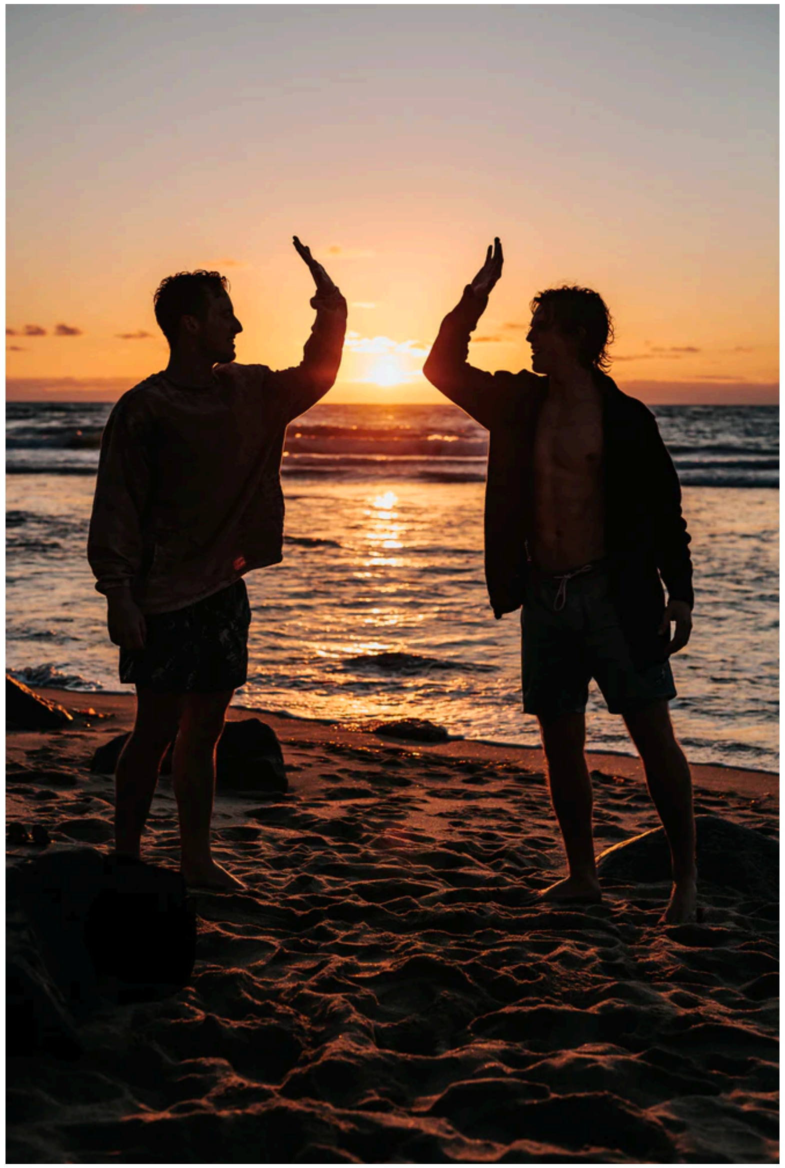


Image source: @jtylernix

# RRS Applications

**(5) Other Emerging Domains**

- **Socializing:** Finding people to meet outside the Internet e.g. to practice a hobby.
- **Skill-sharing:** Sharing skills and learning from other users' shared content.
- **Shared economy:** Peer-to-peer activity, based on collaboration.
- **Mentor-mentee matching**
- **Scientific collaboration**

Image source: @dylandgillis

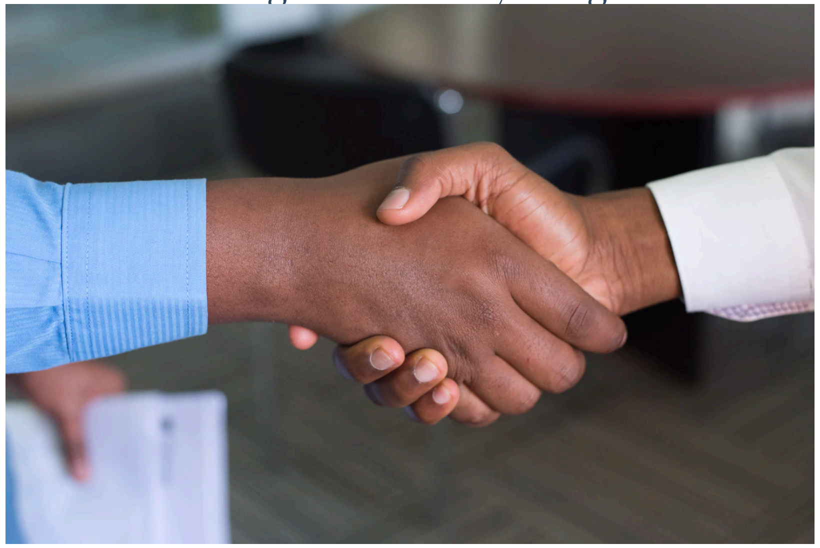


Image source: @cytonn_photography

# Evaluation of RRS

# RRS Evaluation

**Measuring success**

- High cost of *online evaluation* with live users.
- Important to perform a good *offline evaluation* with historical data.

*"For reciprocal recommenders, the evaluation can also use different levels of success. For instance, a job recommendation can be seen as somewhat successful when the user (as a subject) decides to apply for the recommended job; the same recommendation is more successful when the same user is called to be interviewed for the recommended job (the object); and even more successful when the user is selected for the position."*

L. Pizzato, et al. (2013). *Recommending people to people: the nature of reciprocal recommenders with a case study in online dating.* User Modelling and User-Adapted Interaction, 23, pp.447-488.

# RRS Evaluation

**Measuring success**

- Defining a general, realistic success metric requires quantifying the importance of all the factors that may influence such success.
- In a **job recommender**, this measure needs to:
  - Assess the importance of a job seeker being selected for an interview, with respect to…
  - Assessing the importance of applying for a recommended job.
  - Both aspects are important, but maybe not equally.

- The above is difficult to define, hence usually RRS are evaluated using more specific, fine-grained success metrics.

# RRS Evaluation

**Measuring success – evaluation metrics**

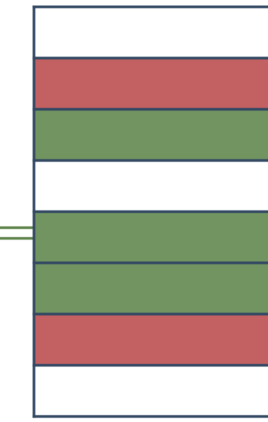
- *Like*: whether a subject user *likes* or not a recommended user.
    - Useful to explain user behavior patterns.
- *Like-back*: whether a subject user *likes* an object user whom likes the subject user → Reciprocated recommendation.

Using either of the above two as *successful* when it occurs, we can define a number of metrics, given a list of *N* recommendations:

**Precision at N:** $\boldsymbol{P@N = \frac{\#successful}{N}}$

$\boldsymbol{P@N = 3/8}$

# RRS Evaluation



**Success rate at N:** $$S@N = \frac{\#successful}{\#successful + \#unsuccessful} = \frac{3}{3+2} = \frac{3}{5}$$

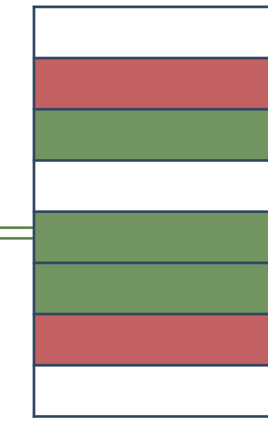
*Unsuccessful* recommendations could include:

- *EoI* sent by the subject user and not replied by the object user.
- *EoI* sent by the subject user and replied negatively.

**Failure rate at N:** $$S@N = \frac{\#unsuccessful}{\#successful + \#unsuccessful} = \frac{2}{3+2} = \frac{2}{5}$$

In some domains like online dating, minimizing failure (false positive rate) might be even more important than maximizing success and precision (true positive rate).

# RRS Evaluation



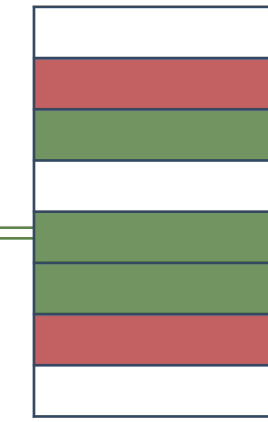
**Recall at N:** How close a recommendation list is to contain all the known successful interactions (ground-truth) involving $x$.

$$\mathbf{R@}\boldsymbol{N} = \frac{\boldsymbol{\#successful}}{\boldsymbol{\#known\ successful\ interactions}}$$

- If test data (interactions to be predicted for $x$) include a total of 6 successful interactions, and three of them have been "predicted" as part of the recommendation list of size $N$=8, then **R@8** = 3/6=**0.5**

**F1 measure:** Combines precision and recall. $F1 = \frac{2\ P \cdot R}{P+R}$

# RRS Evaluation



**Discounted Cumulative Gain at N:**
A measure of ranking quality → It takes the ranking position of recommended users *y* in the recommendation list into account.

$$DCG@N = \sum_{i=1}^{N} \frac{2^{rel_i}-1}{\log_2(i+1)} = \frac{1}{\log_2(3+1)} + \frac{1}{\log_2(5+1)} + \frac{1}{\log_2(6+1)} = 4.12$$

- $rel_i = 1$ if the *i*th recommendation in the list is relevant (successful), and 0 otherwise.

# RRS Evaluation

## GENERAL EVALUATION PIPELINE

1. Split user-user interaction data in the form <*user*$_1$, *user*$_2$, *interaction*> (where *interaction* is a single atomic interaction from *user*$_1$ to *user*$_2$ ) into **training** and **test** data.

# RRS Evaluation

## GENERAL EVALUATION PIPELINE

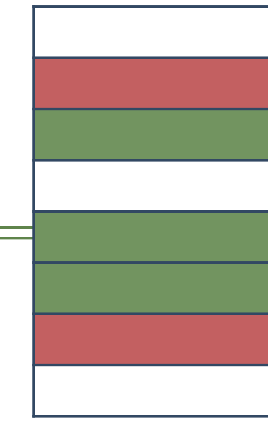
1. Split user-user interaction data in the form <$user_1$, $user_2$, *interaction*> (where *interaction* is a single atomic interaction from $user_1$ to $user_2$ ) into **training** and **test** data.
2. Define what *success* will represent: *liked recommendation* vs *reciprocated recommendation*.

# RRS Evaluation

## GENERAL EVALUATION PIPELINE

1. Split user-user interaction data in the form <*$user_1$*, *$user_2$*, *interaction*> (where *interaction* is a single atomic interaction from *$user_1$* to *$user_2$* ) into **training** and **test** data.
2. Define what *success* will represent: *liked recommendation* vs *reciprocated recommendation*.
3. Select evaluation metric(s) to use – what aspects are relevant in our domain?

# RRS Evaluation



## GENERAL EVALUATION PIPELINE

1. Split user-user interaction data in the form $<user_1, user_2, interaction>$ (where *interaction* is a single atomic interaction from $user_1$ to $user_2$ ) into **training** and **test** data.
2. Define what *success* will represent: *liked recommendation* vs *reciprocated recommendation*.
3. Select evaluation metric(s) to use – what aspects are relevant in our domain?
4. Take a set of tests users *x*, specify a recommendation list *N*, execute our RRS and average for all users the results obtained in the metric(s) used.

# Challenges and future directions

# Common RRS Challenges

*Popularity Bias*

- Popular users in an RRS are those liked by an unduly large of other users in the system.
- If not dealt with properly, the presence of very popular users can negatively affect not only themselves, but also less popular users:
  - **Popular users** may receive an overwhelming number of recommendations, most of which may not be of relevance to them, which may decrease their satisfaction.
  - By contrast, **unpopular users** may be neglected by the RRS, or even suffer a negative experience by being often recommended to popular users who won't reciprocate.

# Common RRS Challenges

*Popularity Bias*

Solutions to manage popular users and prevent 'bias':

- Identify communities of recommendable users around popular users too "split the load".
- Ensure that every user receives as many recommendations as number of times they will be recommended to other people.
- Weighting (balancing) the relative importance of $x$ and $y$ in the calculation of reciprocal preference (as seen earlier).

# Common RRS Challenges

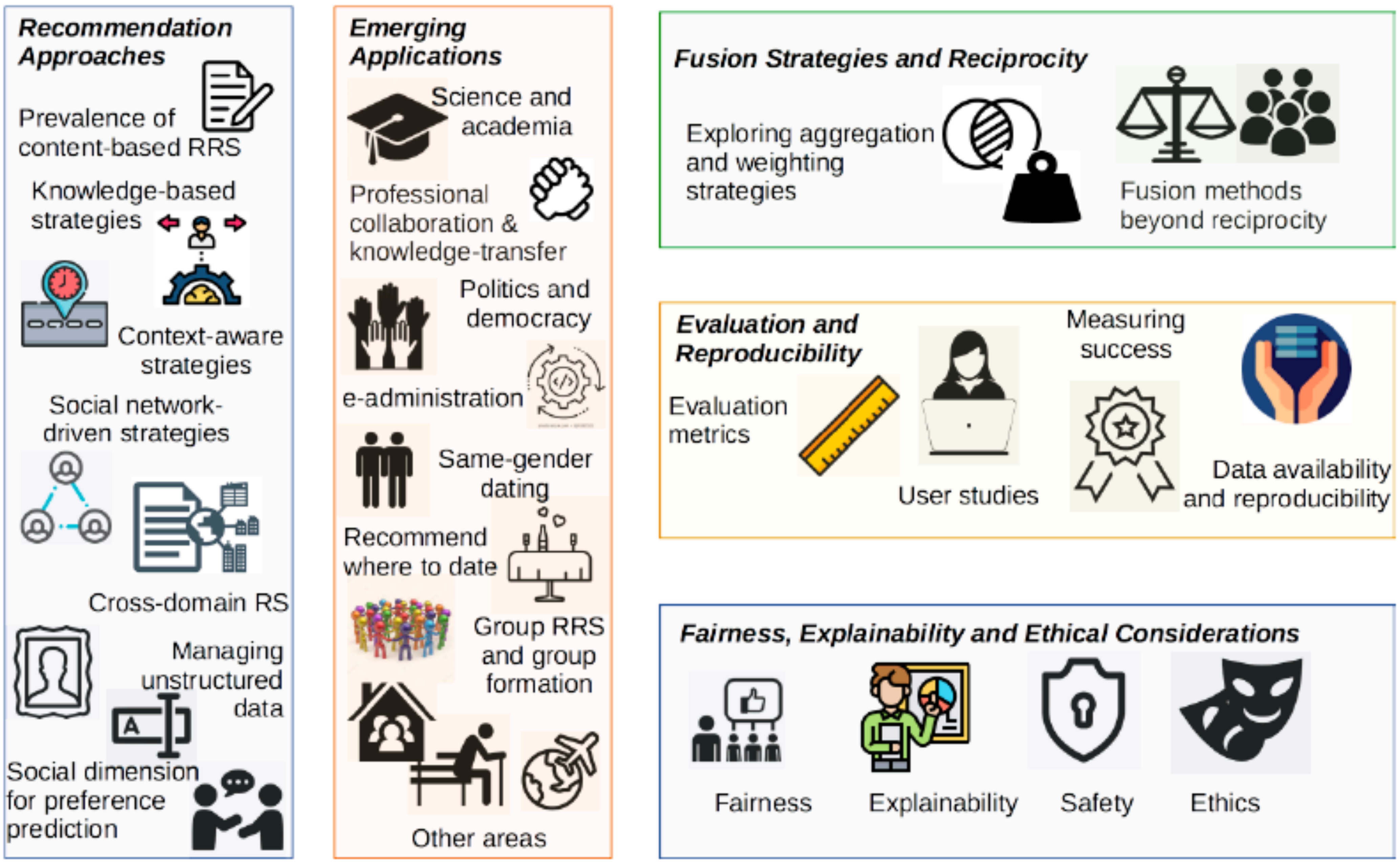
***Fairness and Explainability***

- Ensuring as equal opportunities as possible to all users or communities of them.
- Preventing discrimination, mistreatment and inequality issues, specially with vulnerable groups. (B. Xia et al. 2019, WE-Rec)

- **Explainability** has been barely investigated in RRS. The following work provides a "starting point":

*"Are we able to derive any insight on how these models are learning to recommend relationships? Are attention models able to produce explainable relationship recommendations?" (Tay et al. 2018,CoupleNet)*

# The long way ahead



# Conclusion

## KEY MESSAGES

1. Reciprocal Recommenders recommend people to connect with, introducing the requirement of satisfying the recommended user, not only the end user receiving the recommendation.
2. Online dating is the most prominent application domain, followed by recruitment, online learning and symmetric social networks.
3. Some domains have predominantly content-based approaches, with collaborative filtering and hybrid ones being investigated more recently.
4. The fusion of information (preferences, recommendation lists, etc.) is crucial to measure and integrate reciprocity in recommendation, and increasing success.
5. There are as many pending challenges as opportunities to expand the scope of RRS research.

# Thank you

SIGIR 2020

XI'AN·CHINA

# Reciprocal Recommendation

*Matching Users with the Right Users*

*Contact:* ivanpc(**at**)ugr(**dot**)es

Decision Support and Recommender Systems

Instituto Andaluz Interuniversitario en **Data Science** and **Computational Intelligence**

DaSCI

UNIVERSIDAD DE GRANADA