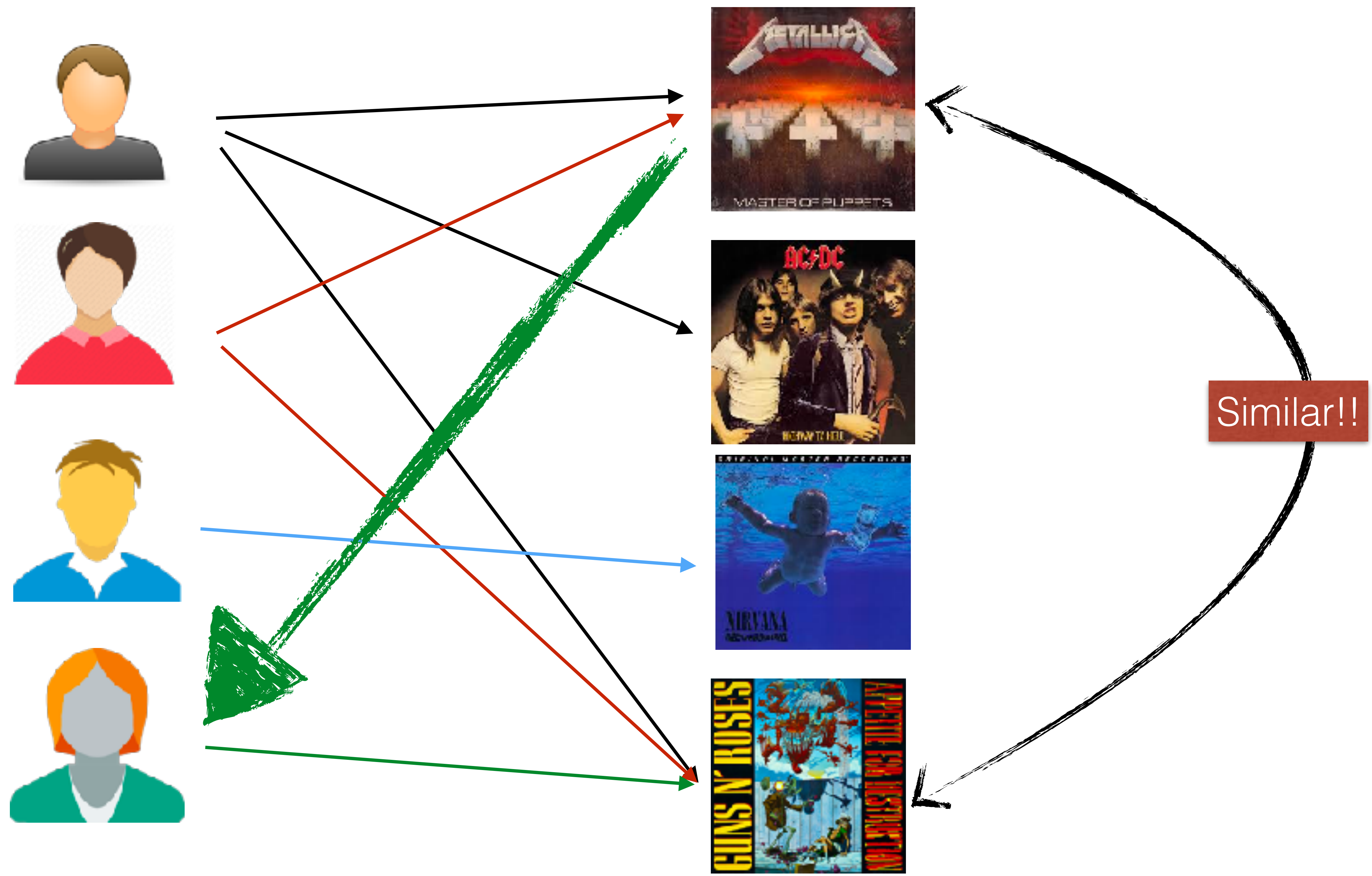
Master on Foundations of Data Science

Recommender Systems

Collaborative Recommender Systems (II)

Santi Seguí | 2017-2018

Item-Based Recommender



Let's see how we can create a **Item-Based CF** for Movie recommendations.

Item-Based Recommender

- Instead on relying on the user similarity, prediction can rely on **item similarities**.
- Item similarity used to be **more stable** than user-similarity. So, the the update frequency of the items similarity is not as critical than user-similarity
  - Item-similarities are more static, while user-similarities are more dynamic

Item-based collaborative filtering recommendation algorithms B Sarwar, G Karypis, J Konstan, J Riedl Proceedings of the 10th international conference on World Wide Web, 285-295	5944	2001

Similarity Measures
What happens with item-based systems?

- Pearson Correlation

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r_a})(r_{b,p} - \bar{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r_b})^2}}$$

- Cosine distance

$$sim(a,b) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Where:

- $sim(a,b)$ is the similarity between user "a" and user "b"
- P is the set of common rated movies by user "a" and "b"
- $r_{a,p}$ is the rating of movie "p" by user "a"
- $\bar{r_a}$ is the mean rating given by user "a"

Are these measures good?

Item Based CF
Pearson Correlation

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r_a})(r_{b,p} - \bar{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r_b})^2}}$$

- Similarities are computed between items.

- Before computing the similarities between columns, each row of the rating matrix is centered to a mean zero.

Adjusted Cosine Similarity

- Computing similarity using basic cosine measure in item-based case has one important drawback: **The differences in rating scale between different users are not taken into account.**

- The Adjusted Cosine Similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_u})^2}}.$$

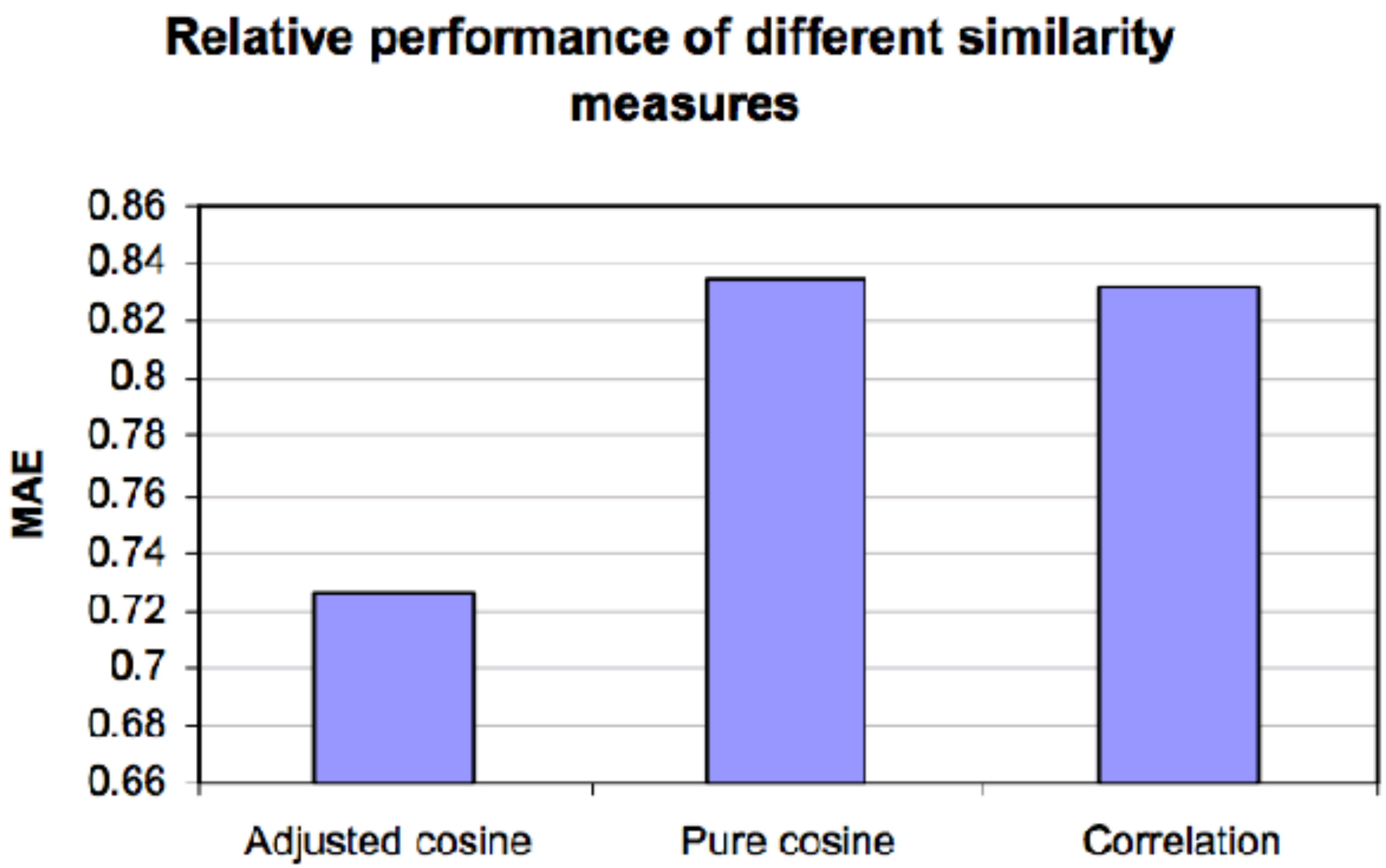


Figure 4: Impact of the similarity computation measure on item-based collaborative filtering algorithm.

Item-based collaborative filtering recommendation algorithms
B Sarwar, G Karypis, J Konstan, J Riedl
Proceedings of the 10th international conference on World Wide Web, 285-295

5944 2001

Item-Based
How do we generate a prediction?

$$\hat{r}_{u,j} = \bar{r}_u + \frac{\sum_{v \in P_u(j)} sim(u,v) \times (r_{v,j} - \bar{r}_v)}{\sum_{v \in P_u(j)} sim(u,v)}$$

Why not another equation?

Exercice:

I want to create a Recommender System for **NETFLIX** using MovieLens dataset.

I have to decide which aproach to use:

a) Non-Personalized

b) User-Based CF

c) Item-Based CF

Plan an implementation plan, think about which is the best under all possibles scenarios you can find.

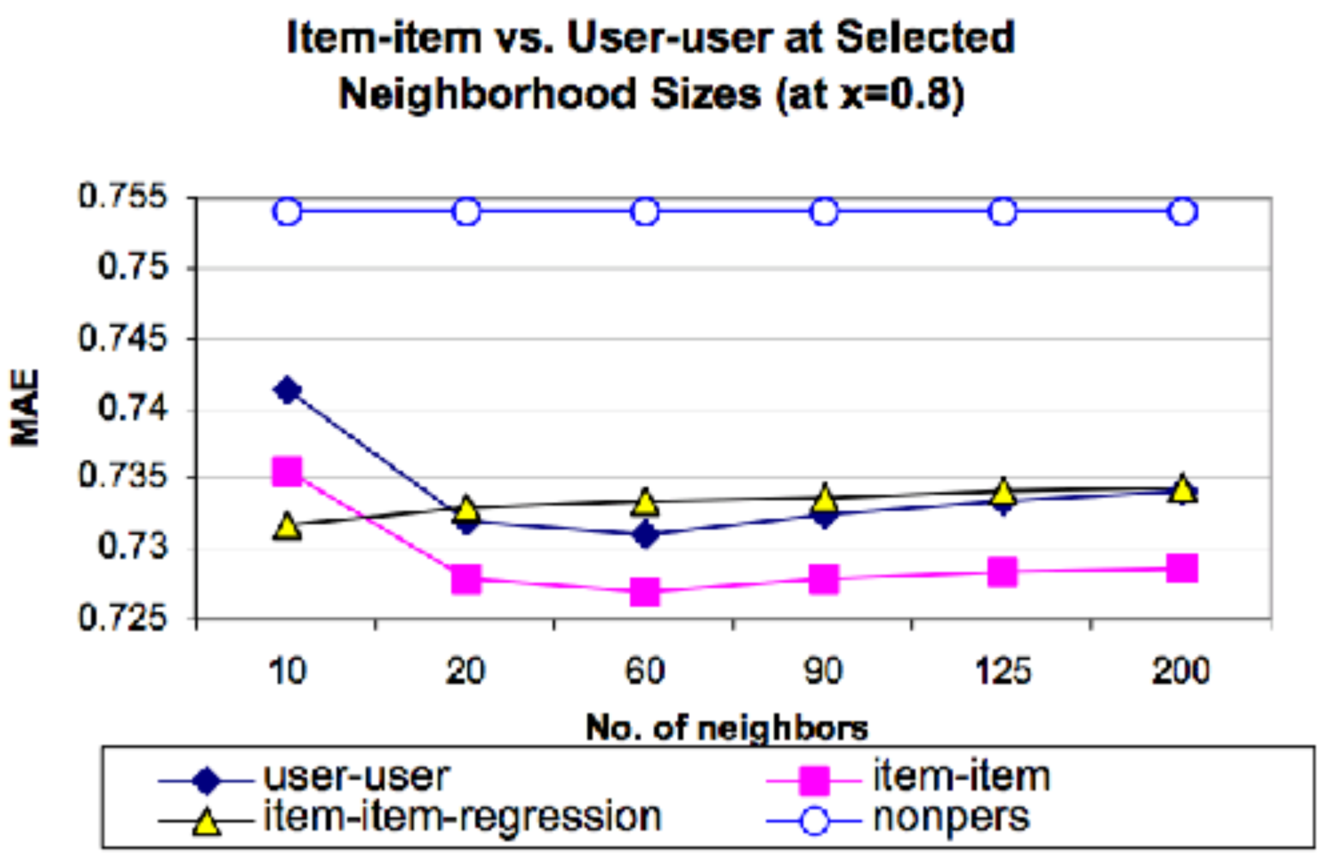
What should we do in order to say which is best?

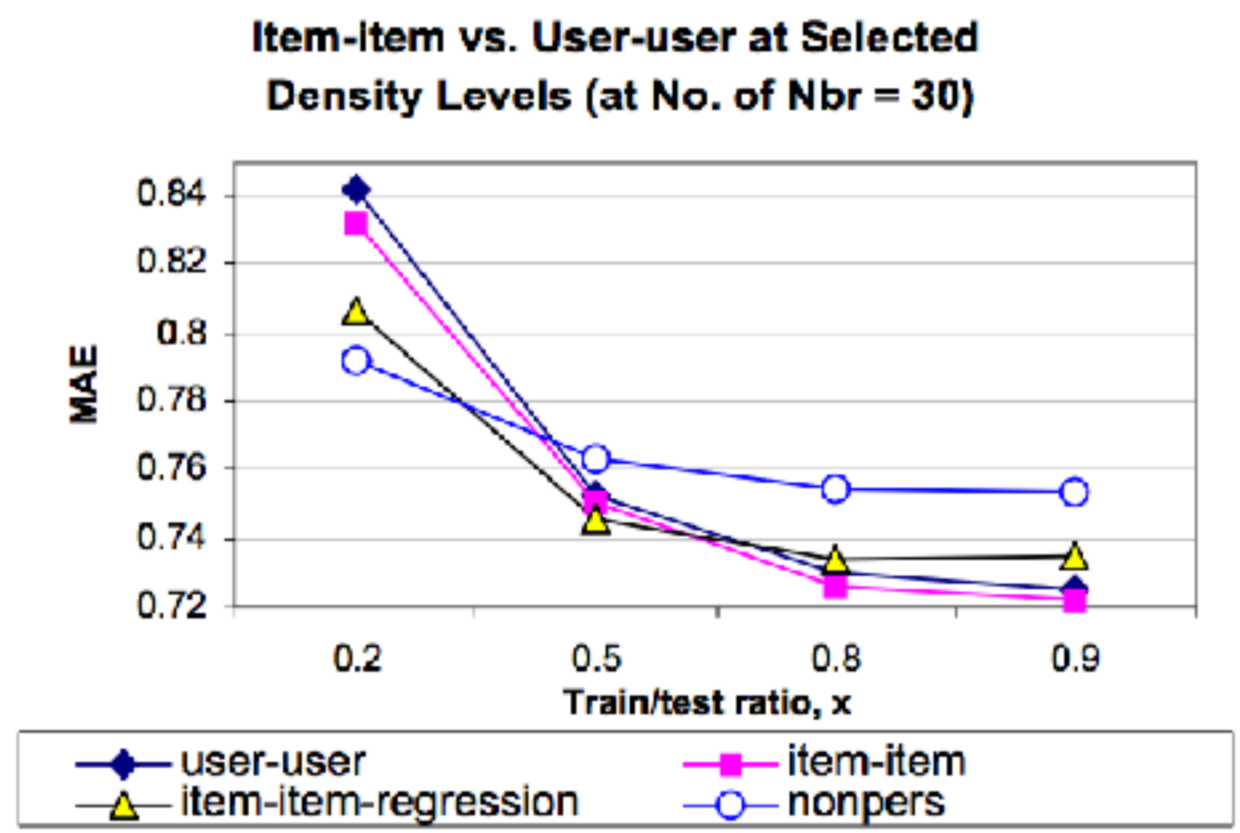
User-Based vs. Item-Based

- m = #users ; n = #items
- Normally, the number of users is much bigger than the number of items.

Computational time:	
O(m^2 n)	O(n^2 m)
Memory Requirements:	
O(m^2)	O(n^2)

User-Based vs. Item-Based





Item-based collaborative filtering recommendation algorithms
B Sarwar, G Karypis, J Konstan, J Riedl
Proceedings of the 10th international conference on World Wide Web, 285-295

5944 2001

User-Based vs. Item-Based

- **Pros User-based**
  - Tend to provide higher diversity (more serendipity)
- **Pros Item-based**
  - Better results (in terms of RMSE)
  - More stable to changes

User-Based vs. Item-Based

	User-Based	Item-Based
Scalability		
Explanation		
Novelty		
Coverage		
Cold start		
Performance		

User-Based vs. Item-Based

	User-Based	Item-Based
Scalability	Bad when #users is huge	Bad when #items is huge
Explanation	Bad	Good
Novelty	Bad	Good
Coverage	Bad	Good
Cold start	Bad for new users	Bad for new items
Performance	Need to get many users history	Only need to get current users's history

Item-Based Nearest Neighbor Regression

- We can replace the (normalized) similarity coefficient AdjustedCosine(j,i) with a unknown parameter w_{ji}^{item} to model the rating prediction of a user u for target item i.

$$\hat{r}_{ui} = \sum_{j \in Q_i(u)} w_{ji}^{item} \cdot r_{uj}$$

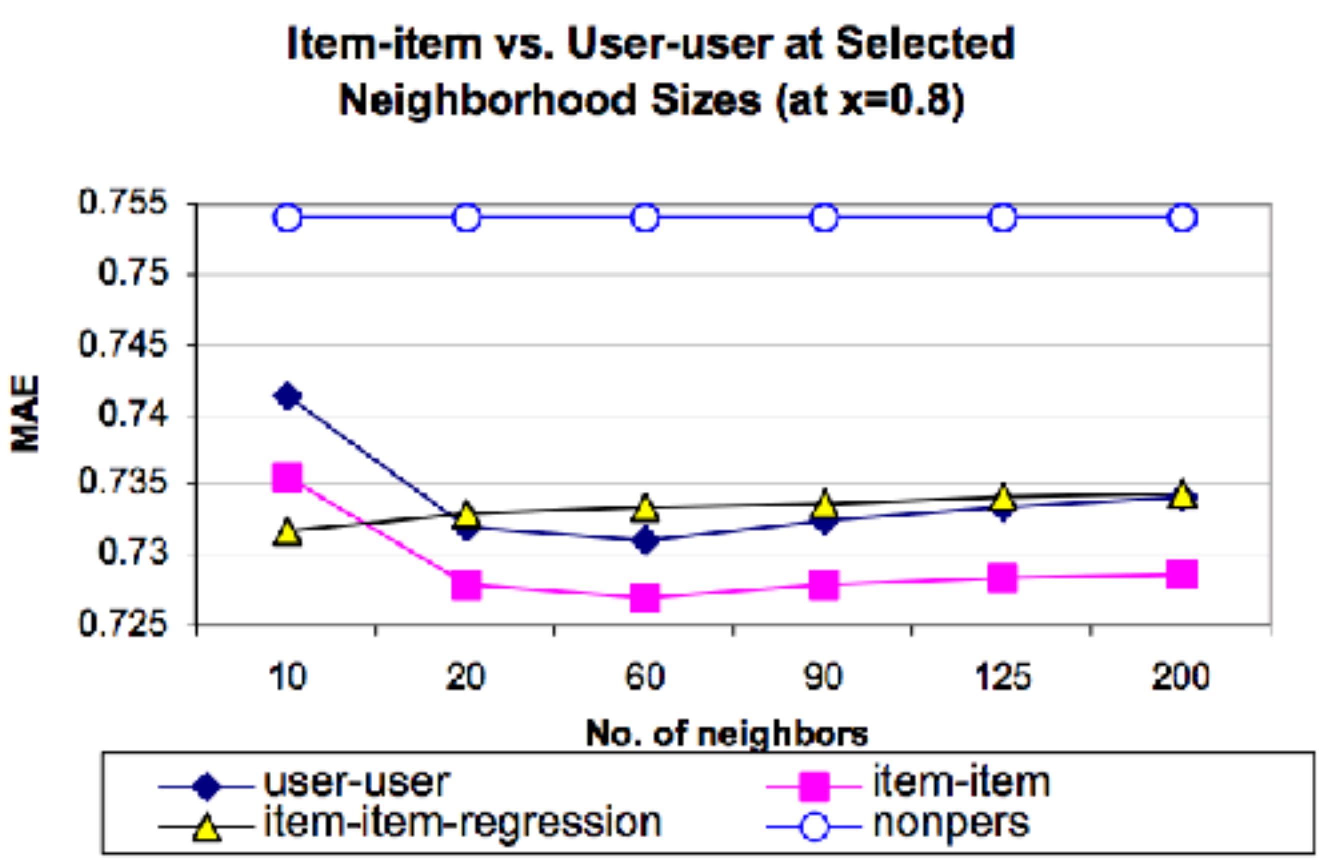
The nearest items in Qi(u) can be determined using the adjusted cosine

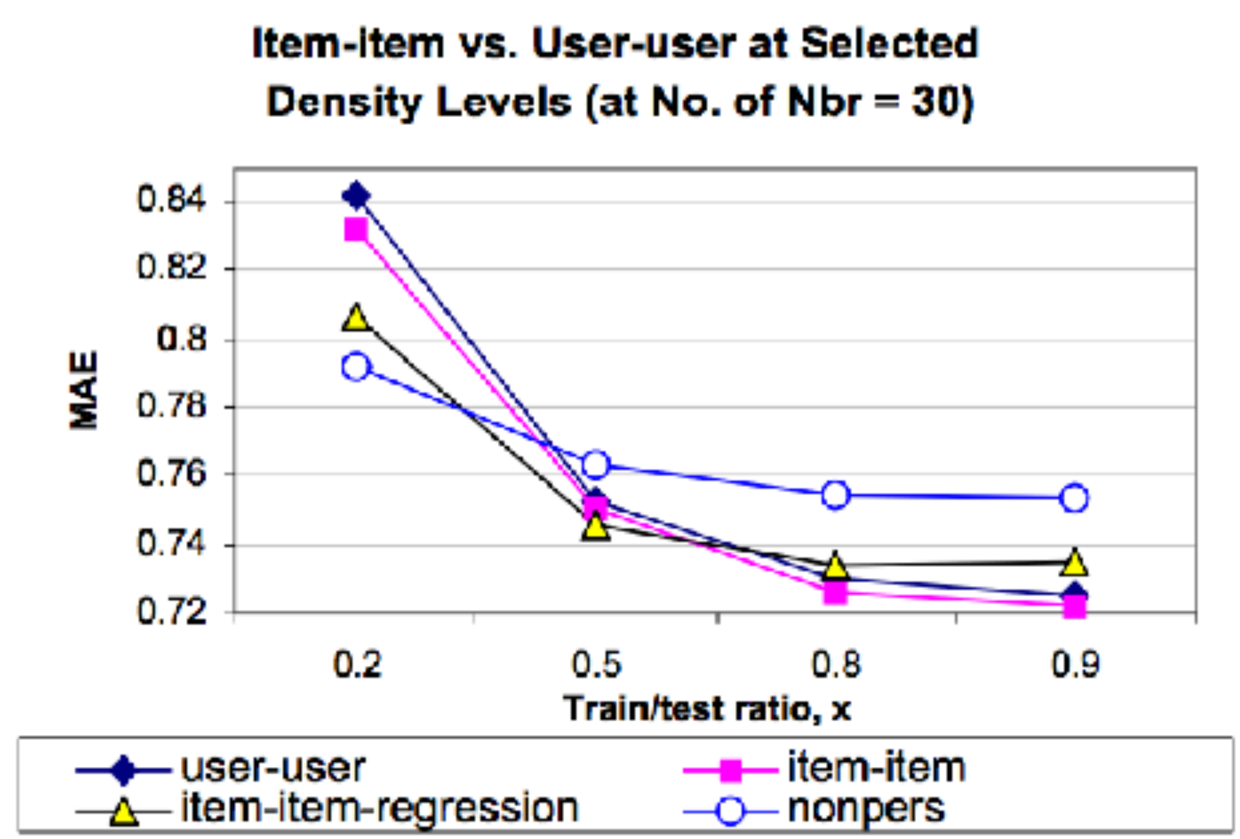
Item-Based Nearest Neighbor **Regression**

$$Minimize\ J_t = \sum_{u \in U_t} (r_{ut} - \hat{r}_{ut})^2$$

$$= \sum_{u \in U_t} \left(r_{ut} - \sum_{j \in Q_t(u)} w_{jt}^{item} \cdot r_{uj} \right)^2$$

User-Based vs. Item-Based





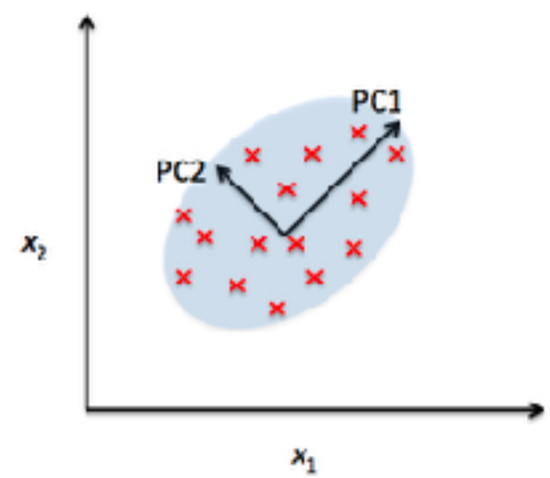
Item-based collaborative filtering recommendation algorithms
B Sarwar, G Karypis, J Konstan, J Riedl
Proceedings of the 10th international conference on World Wide Web, 285-295

5944 2001

Dimensionality Reduction

- Pairwise similarities are hard to robustly be computed in sparse matrices.
- Dimensionality reduction can be used to **improve** neighborhood-based methods both in terms of **quality** and in terms of **efficiency**
- A reduced representation of the data can be created in terms of either row-wise latent factors or in terms of column-wise latent factors.

Dimensionality Reduction



Items

users

m x n **sparse matrix**

≈

Latent Factors

users

m x d **matrix**
where d<<n

Dimensionality Reduction

- The low-dimensional representation can be computed using **PCA** or **SVD-Like** methods.
- After the d-dimensional representation of each user is estimated, the similarity between users can be computed
- Cosine or dot product on the reduced vectors can be used in order to compute the similarity
- More robust since the feature vector is fully specified
- More efficient

Dimensionality Reduction

- How to **obtain** the **d-dimensional representation** on the sparse matrix?
- **SVD Method**. Steps:
    - Augment the *m x n* incomplete rating matrix R -> R_f
        - Mean-user rating or mean-item rating for each row/column
    - Lets define the similarity matrix S as $\mathbf{S = R_f^T R_f}$. S is a positive semi-definite of size *n x n*
    - Determine the dominat basis vectors of R_f by computing the **diagonalization** of the similarity matrix S.
        - $S = P\Lambda P^T$, where P is an *n x n* matrix, whose columns contain the orthonormal eigenvectors of S. Λ is a diagonal matrix containing the non-negative eigenvalues of S along its diagonal.
    - Let denote P_d the *n x d* matrix only containing the columns of P with the largest eigenvalues
    - The low representation of R is obtained by the multiplication of $\mathbf{R_f P_d}$

Dimensionality Reduction

- How to **obtain** the **d-dimensional representation** on the sparse matrix?
- **PCA Method**. Steps:
  - Augment the *m x n* incomplete rating matrix R -> R_f
    - Mean-user rating or mean-item raring for each row/column
  - Lets define the similarity matrix S as **the Covariance Matrix of Rf**
  - Determine the dominat basis vectors of R_f by computing the diagonalization of the similarity matrix S.
    - $S = P \Lambda P^T$, where P is an *n x n* matrix, whose columns contain the orthonormal eigenvectors of S. Λ is a diagonal matrix containing the non-negative eigenvalues of S along its diagonal.
  - Let denote P_d the *n x d* matrix only containing the columns of P with the largest eigenvalues
  - The low representation of R is obtained by the multiplication of $\mathbf{R_f P_d}$

Challenges on Factorization

- Challenges:
  - Missing Values
    - Need a way to fill it
    - Several alternatives, including clever averages and predictions
  - Computational Complexity
  - Lack of transparency/explainability

TASK 1
RecSys Challenge

Create:

1) A recommender system
2) Submit at least one approach
3) Explain your conclusions in class (5 minutes)

You can work with **teams** from **up to 3 members**

Deadline: (around) April 15th (23.55)

Introduction

Twitter is what's happening in the world and what people are talking about right now. On Twitter, live comes to life as conversations unfold, showing you all sides of the story. From breaking news and entertainment to sports, politics and everyday interests, when things happen in the world, they happen first on Twitter.

On the platform, users post and engage with (in the form of Likes, Replies, Retweets and Retweets with comments) content known as "Tweets". This challenge aims to evaluate novel algorithms for predicting different engagement rates at a large scale, and push the state-of-the-art in recommender systems. Following the success and advancements in the domain of top-K recommendations, we aim to encourage the development of new approaches by releasing the largest real-world dataset to predict user engagements. The dataset comprises of roughly 200 million public engagements, along with user and engagement features, that span a period of 2 weeks and contain public interactions (Like, Reply, Retweet and Retweet with comment), as well as 100 million pseudo negatives which are randomly sampled from the public follow graph. While sampling the latter pool of Tweets, we take special care about preserving user privacy.

The submitted methods will be evaluated on a held-out test set generated from more recent Tweets on the platform, and the evaluation metrics will include precision-recall area under curve (PR-AUC) and cross-entropy loss. Participants will also be provided with a validation set, for which the engagement information will be missing. Paying special attention to our users' privacy, the dataset will be updated daily to ensure GDPR-compliance and the corresponding metrics will be updated on the leaderboard.

Prizes

Twitter, as a sponsor of this challenge is providing the dataset, on which all methods will be evaluated. The best three teams will be rewarded with the following prizes:

- Winner: $15,000
- Second team: $10,000
- Third team: $5,000

Dataset description

The Data is available to download here. Fields in each data entry are separated by the 1 character (*0x31 in UTF-8*) and each data entry will be characterized by the following features:

	Feature Name	Feature Type	Feature Description
Tweet Features	Text tokens	*List[long]*	Ordered list of Bert ids corresponding to Bert tokenizati
	Hashtags	*List[string]*	Tab separated list of hastags (identifiers) present in the
	Tweet id	*String*	Tweet identifier
	Present media	*List[String]*	Tab separated list of media types. Media type can be in
	Present links	*List[string]*	Tab separeted list of links (identifiers) included in the T
	Present domains	*List[string]*	Tab separated list of domains included in the Tweet (tw
	Tweet type	*String*	Tweet type, can be either Retweet, Quote, Reply, or Top

trivago

RecSys Challenge 2019 | About the Challenge | Dataset | Leaderboard | Submissions | Profile | Logout

RecSys Challenge 2019

Welcome ACM RecSys Community! For this year's challenge from the online travel domain, build a context-aware accommodation recommendation system that utilises live user interactions.

About the RecSys 2019 Challenge

The goal of the challenge is to use user signals within a session to detect the intent of the user and to update the recommendation of accommodations provided to the user. Given a dataset of the interactions of the users on our website and metadata for the items they interacted with, the participants are tasked with predicting what items have been clicked in the later part of a session.

More about the challenge →
Dataset →

Current Leaderboard

Team	Score
CustomerSuccess	0.3713
trivago	0.288448
Sharknado	0.288448
Grubhub Personalization	0.288448
Team Buctù	0.288448

UNIVERSITAT DE BARCELONA

DASW