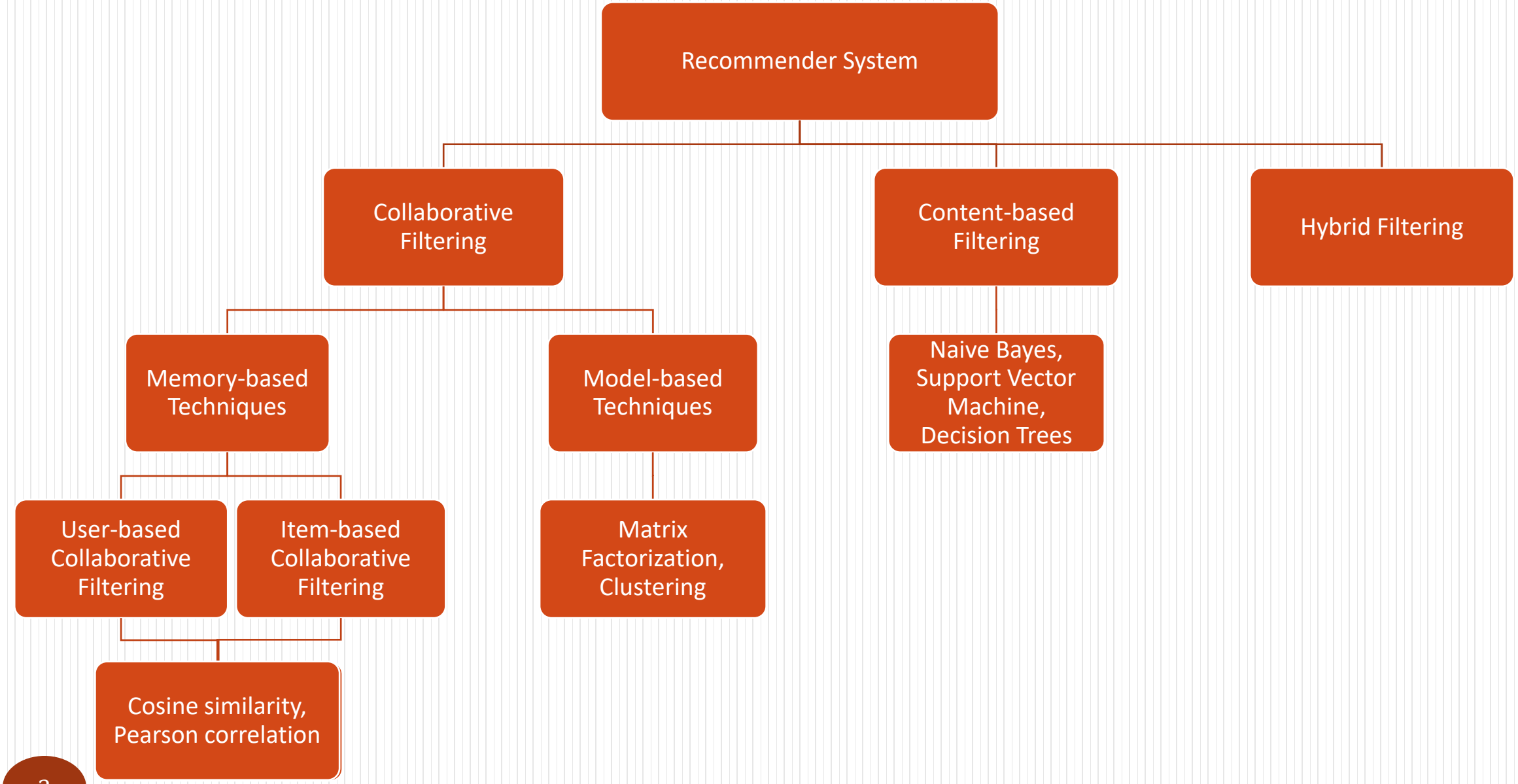
Improving the Performance of Recommendation Systems

Eyad Kannout
University of Warsaw
MIMUW
Intelligent Systems Seminar

Agenda

- Introduction to Recommendation Systems
- Context Clustering-based Recommender Systems (CoCl)
- Overview for Factorization Machines and Association Rules
- FMAR Recommender System
- Experiments/Evaluation methodology
- Results
- Future work


Recommender System
Collaborative Filtering
Content-based Filtering
Hybrid Filtering
Memory-based Techniques
Model-based Techniques
Naive Bayes, Support Vector Machine, Decision Trees
User-based Collaborative Filtering
Item-based Collaborative Filtering
Matrix Factorization, Clustering
Cosine similarity, Pearson correlation

Collaborative Filtering

- user-item matrix (V) =

$$\begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{ccccccc} p_1 & p_2 & \cdots & p_j & \cdots & p_m \end{array} \\ \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,j} & \cdots & v_{1,m} \\ v_{2,1} & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & v_{i,j} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ v_{n,1} & \cdots & \cdots & \cdots & \cdots & v_{n,m} \end{bmatrix} \end{array} = \begin{bmatrix} 0 & 0 & 4 & 5 & 0 & 1 \\ 0 & 3 & 4 & 0 & 1 & 2 \\ 0 & 2 & 3 & 4 & 3 & 2 \\ 2 & 0 & 0 & 1 & 2 & 4 \\ 2 & 0 & 5 & 4 & 3 & 2 \end{bmatrix}$$

Memory-based Techniques

- Cosine Similarity: $\cos(u_i, u_k) = \dfrac{\sum_{j=1}^{m} v_{ij} v_{kj}}{\sqrt{\sum_{j=1}^{m} v_{ij}^2 \sum_{j=1}^{m} v_{kj}^2}}$

- Pearson Correlation: $S(u_i, u_k) = \dfrac{\sum_j (v_{ij} - \bar{v}_i)(v_{kj} - \bar{v}_k)}{\sqrt{\sum_j (v_{ij} - \bar{v}_i)^2 \sum_j (v_{kj} - \bar{v}_k)^2}}$

Where:

- v_{ij}: is the rating that the user u_i gave to the product p_j
- $v`_i$: is the mean rating given by the user u_i

- V = $\begin{bmatrix} 0 & 3 & 4 & 5 & 2 & 3 \\ 0 & 3 & 4 & 4 & 0 & 2 \\ 0 & 2 & 3 & 4 & 1 & 4 \\ 2 & 0 & 0 & 1 & 3 & 4 \\ 2 & 0 & 5 & 4 & 3 & 3 \end{bmatrix}$ $\xrightarrow{Pearson\ Correlation}$ $\begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{ccccc} u_1 & u_2 & \dots & u_i & u_n \end{array} \\ \begin{bmatrix} 1 & S(1,2) & \dots & S(1,i) & S(1,n) \\ & 1 & \dots & \dots & S(2,n) \\ & & 1 & \dots & \dots \\ & & & 1 & \dots \\ & & & & 1 \end{bmatrix} \end{array}$ = Similarity Matrix(S_{n*n})

Model-based Techniques

Singular Value Decomposition (SVD):

$$X_{n\times m} = U_{n\times r} \times S_{r\times r} \times V^t_{r\times m}$$

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & & & \\ \vdots & & \ddots & \vdots \\ x_{n1} & & \dots & x_{nm} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1r} \\ u_{21} & & & \\ \vdots & & \ddots & \vdots \\ u_{n1} & & \dots & u_{nr} \end{bmatrix} \begin{bmatrix} s_{11} & 0 & \dots & 0 \\ & s_{21} & & \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & s_{rr} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1m} \\ v_{21} & & & \\ \vdots & & \ddots & \vdots \\ v_{r1} & & \dots & v_{rm} \end{bmatrix}$$

Context Clustering-based Recommender Systems (CoCl)

- A hybrid model that utilizes the contextual information and KMeans clustering algorithm to create new forms of user-item matrices.
- Applying the traditional collaborative filtering approach on these new matrices produces more accurate results.
- CoCl provides two approaches:
  - **RateClust:** the ratings in the utility matrix will be grouped in such a way that the ratings with similar contextual information will be together.
  - **UserClust:** the users in the utility matrix will be grouped based on their ratings in dedicated contexts.

RateClust

- We group the ratings that are given in similar contexts.
- The contextual information in our dataset describes the situation in which the user consumed/rated the item.

Rating	Time	Location	Mood	Social
4.5	morning	Home	Negative	Alone
4	Night	Public place	Neutral	Friends
. .	. .	. .	. .	
3	morning	Public place	Positive	My partner
5	Evening	Friend's house	Neutral	My family

UserClust

- We group the users that share the same behavior in similar contexts.
- IF the mood context (positive, neutral, negative, missing) is selected: Then for each user, we calculate the average rating given in every mood possible value.

UserId	Avg positive rating	Avg neutral rating	Avg negative rating	Avg unknown rating
1	4.07	3.83	3.50	4.31
2	4.00	4.25	5.00	3.00
. .	. .	. .	. .	
20	3.80	3.76	3.89	4.46
30	4.20	4.20	3.00	4.00

New Forms of User-Item Matrix

- The ratings are aggregated for each movie based on the cluster they belong to.
- The new generated matrices can be utilized in different ways while building the recommender system:
  - Divide the aggregated user-item matrix into smaller matrices based on the cluster the records belong to.
  - Build one recommender system without dividing the aggregated user-item matrix.

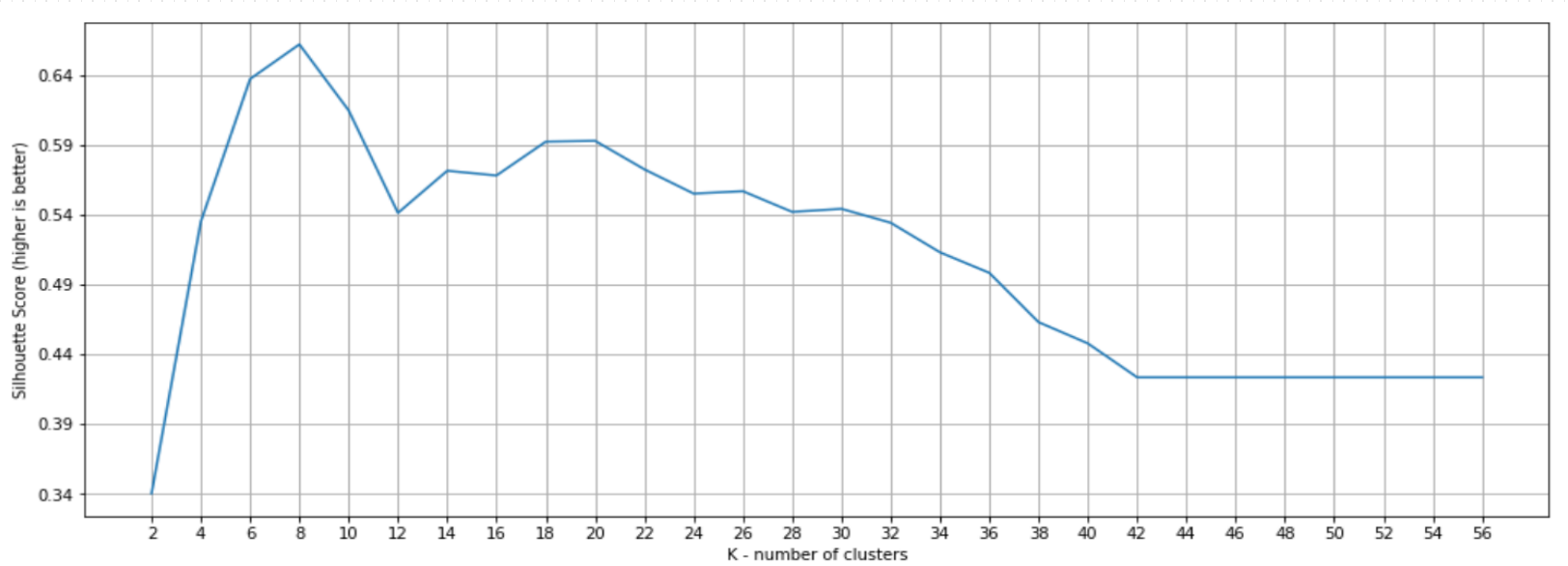
Context variables in LDOS-CoMoDa dataset

time	Morning, Afternoon, Evening, Night
daytype	Working day, Weekend, Holiday
season	Spring, Summer, Autumn, Winter
location	Home, Public place, Friend's house
weather	Sunny / clear, Rainy, Stormy, Snowy, Cloudy
social	Alone, My partner, Friends, Colleagues, Parents, Public, My family
endEmo	Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral
dominantEmo	Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral
mood	Positive, Neutral, Negative
physical	Healthy, Ill
decision	User decided which movie to watch, User was given a movie
interaction	first interaction with a movie, n-th interaction with a movie

Optimal Number of Clusters

- Silhouettes score to select the optimal number of clusters for both versions of CoCl, RateClust and UserClust.

$$SilhouetteScore = (x - y)/max(x, y)$$



Calculating mean silhouettes score over all samples for different number of clusters

Evaluation Metrics

- To measure and compare the performance of various recommendation models we use:
  - *RMSE:* imposes a penalty over the larger errors:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \grave{x_i})^2}$$

  - *MAE*: measures the average magnitude of the errors in a set of predictions, without considering their direction:

$$\frac{1}{N}\sum_{i=1}^{N}\left|x_i - \grave{x_i}\right|$$

Performance Comparison and Analysis

- The recommendation systems produced by CoCl will be evaluated using four methods:
  - Cross-validation method
  - Holdout evaluation method
  - Building multiple recommender systems based on generated clusters
  - Ensemble recommender systems

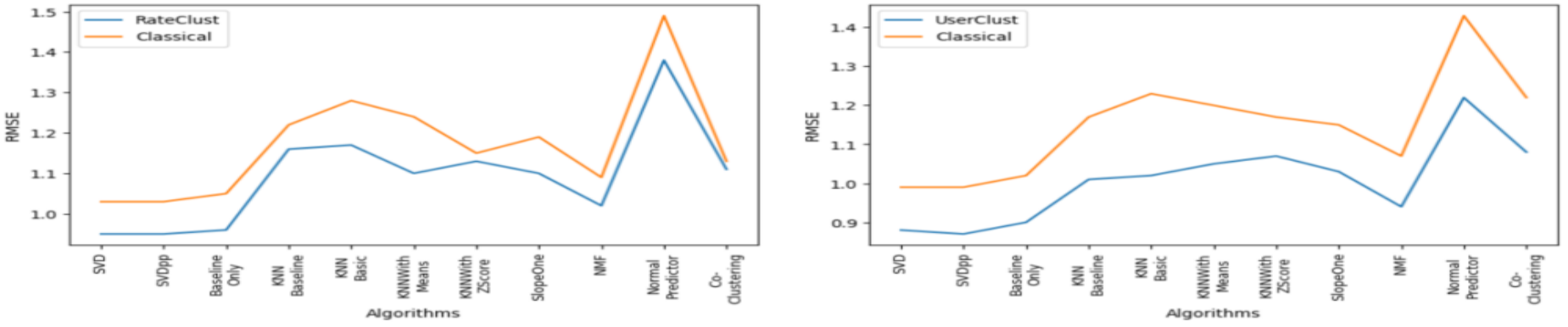
Cross-Validation Method

TABLE II: Cross Validation - Rating-based clustering VS Classical

Model	Metric	SVD	SVDpp	Baseline Only	KNN Baseline	KNN Basic	KNN WithMeans	KNN WithZScore	Slope One	NMF	Normal Predictor	Co-Clustering
RateClust	RMSE	0.96	0.95	0.95	1.07	1.08	1.08	1.09	1.06	1.03	1.36	1.10
	MAE	0.76	0.76	0.76	0.78	0.79	0.82	0.82	0.82	0.83	1.08	0.83
Classical	RMSE	1.01	1.01	1.02	1.14	1.18	1.16	1.12	1.14	1.09	1.41	1.14
	MAE	0.81	0.81	0.82	0.87	0.89	0.89	0.85	0.89	0.88	1.13	0.88

TABLE III: Cross Validation - User-based clustering VS Classical

Model	Metric	SVD	SVDpp	Baseline Only	KNN Baseline	KNN Basic	KNN WithMeans	KNN WithZScore	Slope One	NMF	Normal Predictor	Co-Clustering
UserClust	RMSE	0.95	0.94	0.96	1.13	1.13	1.08	1.08	1.09	1.01	1.33	1.13
	MAE	0.75	0.74	0.76	0.84	0.84	0.80	0.81	0.84	0.81	1.04	0.86
Classical	RMSE	0.99	0.99	1.03	1.17	1.23	1.20	1.17	1.15	1.06	1.53	1.18
	MAE	0.80	0.80	0.84	0.90	0.93	0.93	0.91	0.91	0.87	1.21	0.91



(a) Rating-based clustering VS Classical

(b) User-based clustering VS Classical

Fig. 2: Generalization assessment using testing set

Holdout Evaluation Method

TABLE IV: Performance Comparison - Rating-based clustering (one recommender system) VS Classical

Model	Metric	SVD	SVDpp	Baseline Only	KNN Baseline	KNN Basic	KNN WithMeans	KNN WithZScore	Slope One	NMF	Normal Predictor	Co-Clustering
RateClust	RMSE	0.86	0.85	0.86	1.02	1.03	1.02	1.06	0.98	0.92	1.31	0.99
	MAE	0.69	0.68	0.68	0.72	0.73	0.78	0.79	0.75	0.73	1.07	0.76
Classical	RMSE	0.94	0.94	0.95	1.11	1.08	1.06	1.08	1.05	1.00	1.38	1.11
	MAE	0.76	0.77	0.76	0.83	0.83	0.80	0.83	0.82	0.80	1.12	0.86

TABLE V: Performance Comparison - User-based clustering (one recommender system) VS Classical

Model	Metric	SVD	SVDpp	Baseline Only	KNN Baseline	KNN Basic	KNN WithMeans	KNN WithZScore	Slope One	NMF	Normal Predictor	Co-Clustering
UserClust	RMSE	0.89	0.89	0.89	0.95	0.96	1.00	1.05	0.96	0.98	1.34	0.97
	MAE	0.69	0.70	0.70	0.68	0.68	0.75	0.78	0.75	0.76	1.07	0.73
Classical	RMSE	1.04	1.03	1.05	1.12	1.18	1.15	1.14	1.17	1.14	1.49	1.18
	MAE	0.84	0.84	0.85	0.87	0.89	0.89	0.88	0.92	0.92	1.22	0.93

Multiple Recommender Systems based on Clusters

TABLE VI: Performance Comparison - Rating-based clustering (Multiple recommender systems) VS Classical

Model	Metric	SVD	SVDpp	Baseline Only	KNN Baseline	KNN Basic	KNN WithMeans	KNN WithZScore	Slope One	NMF	Normal Predictor	Co-Clustering
RateClust	RMSE	0.78	0.79	0.78	0.82	0.83	0.80	0.80	0.86	0.82	1.17	0.86
	MAE	0.61	0.62	0.60	0.61	0.62	0.56	0.56	0.66	0.62	0.95	0.67
Classical	RMSE	0.94	0.93	0.95	1.08	1.12	1.08	1.06	1.05	1.00	1.35	1.06
	MAE	0.77	0.76	0.76	0.83	0.83	0.83	0.80	0.82	0.80	1.06	0.83

TABLE VII: Performance Comparison - User-based clustering (Multiple recommender systems) VS Classical

Model	Metric	SVD	SVDpp	Baseline Only	KNN Baseline	KNN Basic	KNN WithMeans	KNN WithZScore	Slope One	NMF	Normal Predictor	Co-Clustering
UserClust	RMSE	0.88	0.89	0.88	0.89	0.90	0.88	0.89	0.93	1.04	1.26	0.91
	MAE	0.68	0.69	0.69	0.64	0.64	0.60	0.61	0.71	0.82	1.00	0.68
Classical	RMSE	1.04	1.03	1.05	1.12	1.18	1.15	1.14	1.17	1.14	1.39	1.15
	MAE	0.85	0.83	0.85	0.87	0.89	0.89	0.88	0.92	0.92	1.12	0.92

Ensemble Recommender Systems

- We create ensemble recommender systems for RateClust, UserClust and classical models.
- The main idea is to aggregate the ratings produced by each algorithm in order to produce the final ratings in the target recommender system.
- We select the best three algorithms that produce the most accurate results in previous evaluation methods (SVD, KNNBaseline and BaselineOnly).
- While building the clustering-based recommender systems; the entire aggregated dataset is used without any splitting.

Ensemble Recommender Systems

TABLE VIII: Rating-based clustering VS Classical (Ensemble Recommender System)

model	Ensemble Recommender System	
	RMSE	MAE
RateClust	0.95	0.74
Classical	1.03	0.79

TABLE IX: User-based clustering VS Classical (Ensemble Recommender System)

model	Ensemble Recommender System	
	RMSE	MAE
UserClust	1.00	0.80
Classical	1.08	0.86

Improving Recommendation Speed Using Association Rules

- We focus on speeding up the process of generating the recommendations without impacting the accuracy.
- In this model we combine two approaches in order to speed up the recommendation systems.
- It is based on factorization machines and association rules (FMAR).

Why Factorization Machines

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i$$

$$\widehat{y}(x) = w_0 + \sum_{i=1}^{N} w_i x_i + \sum_{i=1}^{N} \sum_{i=1+1}^{N} w_{ij} x_i x_j$$

Poly2: $w_{i,j}$

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

Linear Regression — Feature Interactions

- Factorization Machines Trick!
  - It finds latent vectors for each feature and compute the weight of feature interactions as a dot product of those vectors.

Why Association Rules

- The basic idea of association rules is to uncover all relationships between elements.
- Association Rules Vs Collaborative Filtering.
  - Association Rules: all transactions are studied as one group.
  - Collaborative Filtering: transaction are grouped by userID.
- An association rule consists of antecedent and consequent.
- Various metrics exist to identify the most important rules and calculate their strength, such as support, confidence and lift.

Support

- This measure gives an idea of how frequent an itemset is in all the transactions.
- Value of support helps us identify the rules worth considering for further analysis

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

$$\text{supp}(A \Rightarrow 0) = P(A \wedge 0) = P(A)P(0 \mid A) = P(0)P(A \mid 0)$$
$$\text{supp}(B \Rightarrow 1) = P(B \wedge 1) = P(B)P(1 \mid B) = P(1)P(B \mid 1)$$

Confidence

- This measure is an indication of how often the rule has been found to be true.
- Confidence (X→Y), with respect to a set of transactions T, is the proportion of the transactions that contain X which also contain Y.

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

$$\mathrm{conf}(X \Rightarrow Y) = \mathrm{supp}(X \cup Y)/\mathrm{supp}(X)$$

$$\mathrm{conf}(A \Rightarrow 0) = P(0 \mid A)$$
$$\mathrm{conf}(B \Rightarrow 1) = P(1 \mid B)$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Lift

- This measure is used to discover (exclude) the weak rules that have high confidence.
- Mathematically, lift can be calculated by dividing the confidence by the unconditional probability of the consequent

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

$$\text{lift}(A \Rightarrow 0) = \frac{P(0 \mid A)}{P(0)} = \frac{P(A \wedge 0)}{P(A)P(0)}$$
$$\text{lift}(B \Rightarrow 1) = \frac{P(1 \mid B)}{P(1)} = \frac{P(B \wedge 1)}{P(B)P(1)}$$

$$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X) \times \text{supp}(Y)}$$

FMAR Model

- A hybrid model that utilizes factorization machines and Apriori algorithm to minimize the prediction latency of recommender system.
- Steps:

Algorithm 1 FMAR Model

for all users **do**
    Find related association rules
    Create user profile which recommends a set of items
    Filter the items that are passed to recommendation engine
end for

How Association Rules Are Generated

Algorithm 2 FMAR Model - Association Rules Generation

```
Extract favorable reviews                                  ▷ ratings > 3
Find frequent item-sets                          ▷ support > min_support
Extract all possible association rules
Compute confidence and lift for every rule
if (confidence < min_confidence) or (lift < min_lift) then
    Filter out the rules
end if
Create users' profile
```

Creating Users' Profile

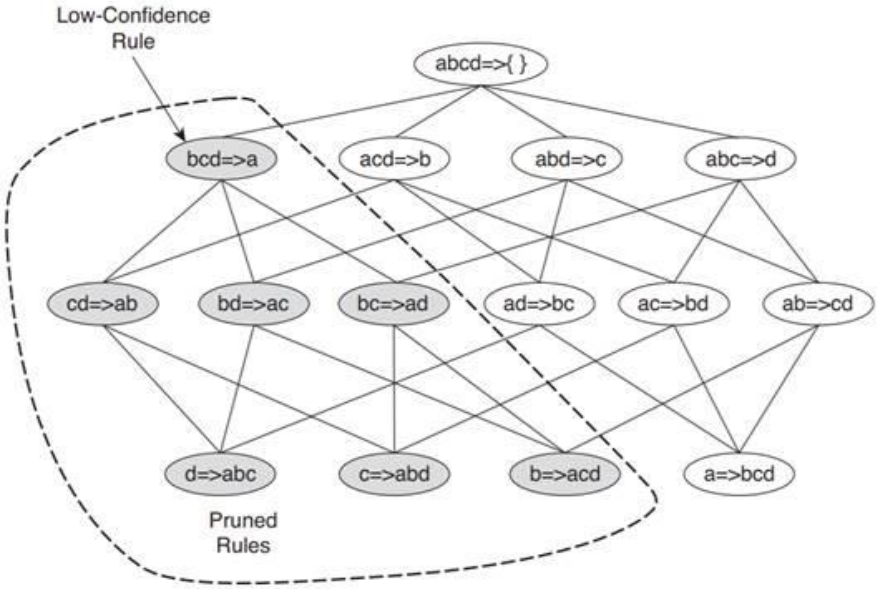
Algorithm 3 FMAR Model - Users' Profile Generation

```
for all users do
    Find high rated items based on the rating history
    Find association rules that their antecedents are subset of high rated items
    Recommend all consequences of these rules
end for
```

- For example:
  - User1: highly rated {2, 5, 8, 10} items.
  - {5} –> {20}, {2, 5} –> {30}, {8} –> {40}, {2, 5, 8, 10} –> {70}

Anti-Monotone Property

- If we drop out an item from an itemset, support value of new itemset generated will either be the same or will increase.
- All subsets of a frequent itemset must also be frequent.
- This property is considered while calculating support and confidence.



Efficiency of FMAR

- When we try to find new frequent item-set, there is no need to check all items and calculate support for every new combination.
- Assume we need to find new frequent item-set based on {A,B,C}, if item D does not form frequent item-set with {A,B}, then it will not form frequent item-set with {A,B,C}.
- Main idea here is to find intersections for all items that produce frequent item-sets with {A,B}, {A,C}, {B,C}.

Evaluation Methodology

- Dataset: MovieLens 100K dataset is a stable benchmark dataset which consists of 1682 movies and 943 users who provide 100,000 ratings on a scale of 1 to 5.
- It is important to note that in this paper, we are not concerned about users' demographics and contextual information since the association rules are created based only on rating history.
- In order to generate predictions, we employ a factorization machines model which is created using the publicly available software tool libFM.

Parameters Selection

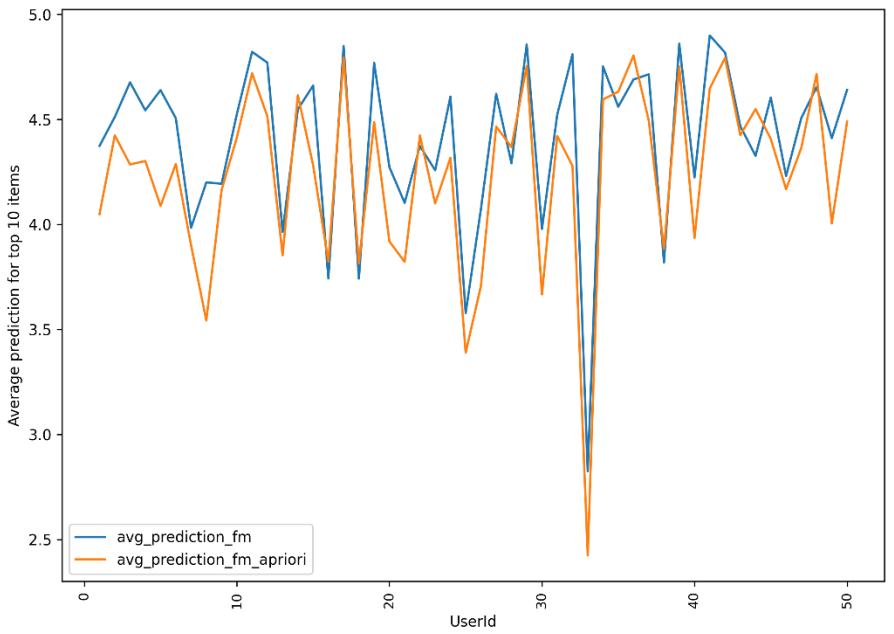
- Several experiments are conducted in order to select the appropriate values of parameters in previous algorithm, such as min_support and min_confidence.
- Multiple factors are taken into consideration while selecting those values, including accuracy, number of generated rules, and memory consumption.

Performance Comparison and Analysis

- Several methods are used to compare between FMAR and FM recommender system.
- In every method, we create two sets of items for every user:
  - Original set which contains all items that are not rated before by the user.
  - Short-listed set which is created by filtering the original set using the association rules.
  - We pass both sets to factorization machines model to generate predictions.
  - We arrange the results in descending order based on the predicted values, and find the top 10 items for every set.
  - Compare the results.

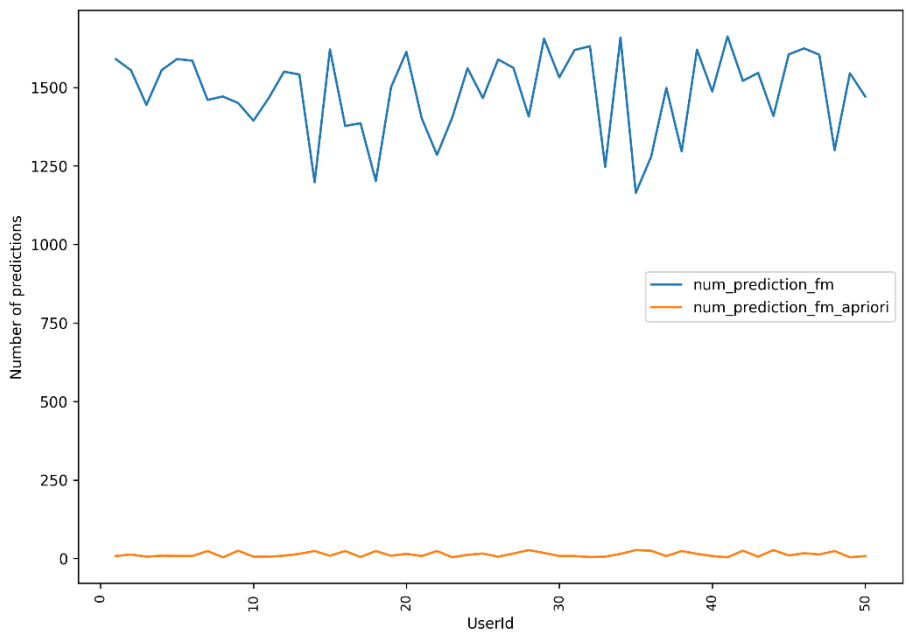
First Method

- We calculate the average of prediction for the top 10 items which are generated in both recommendation engines.
- The main goal of this approach is to make sure that accuracy of predicted items is not highly impacted after filtering the items using the association rules.



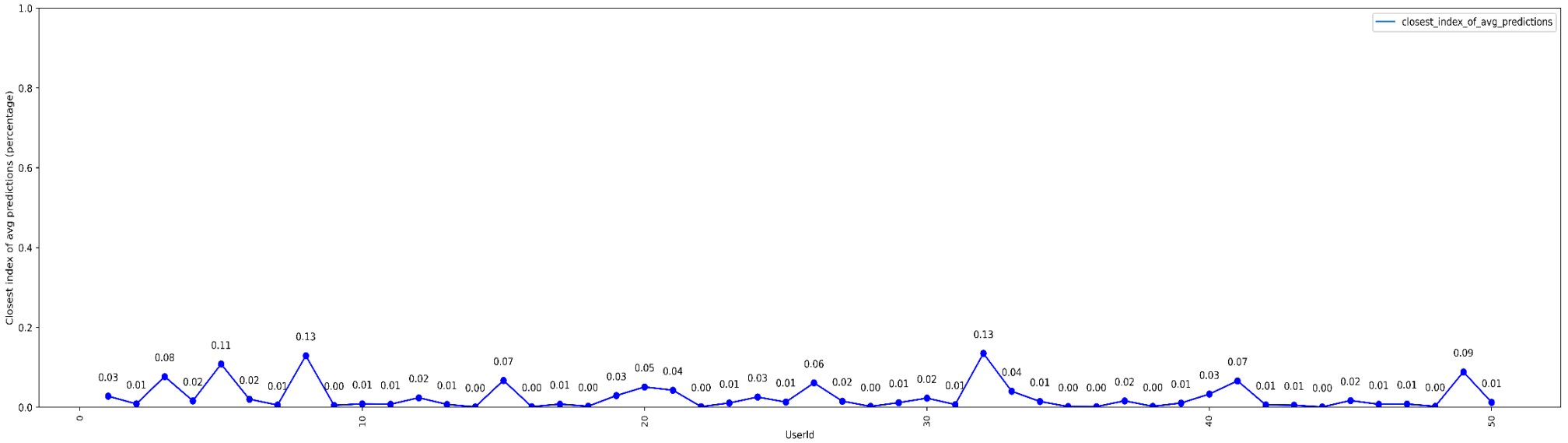
Second Method

- We compare between the number of items in original and short-listed sets.
- The main idea here is to show how many items we have to pass to factorization machines model before and after using the association rules.



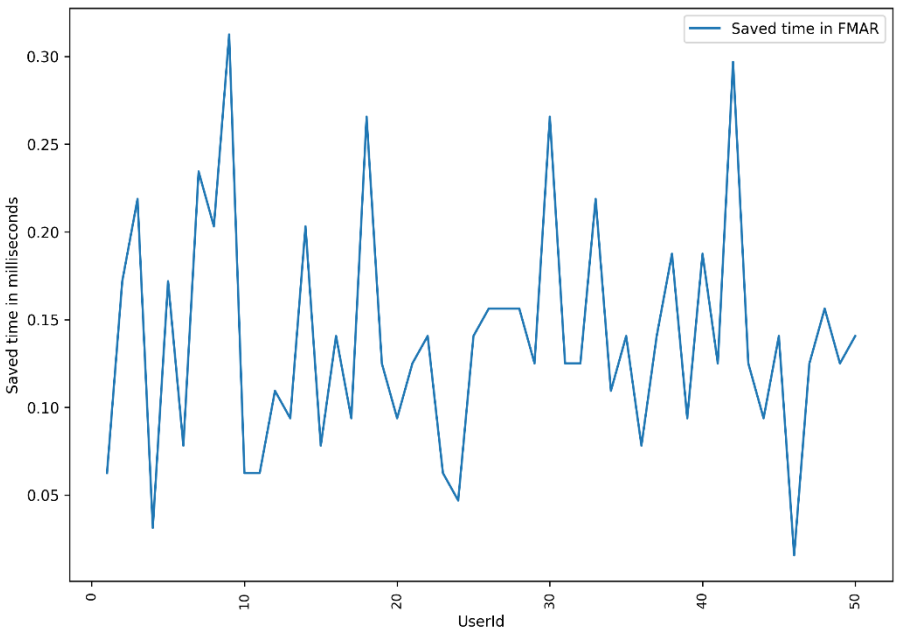
Third Method

- We calculate the average of predictions for the top 10 items in the short-listed set. Then, we find the closest index of this average value in the original set.
- The original set here is sorted in descending order based on the predicted value for each item.



Fourth Method

- We compare between FMAR and classical recommender system in terms of the elapsed time necessary to make a prediction.
- The saved time can be increased based the length of original set of items. So, we are expecting to save more time when we use larger dataset



Thank you

Q&A

eyad.kannout@mimuw.edu.pl