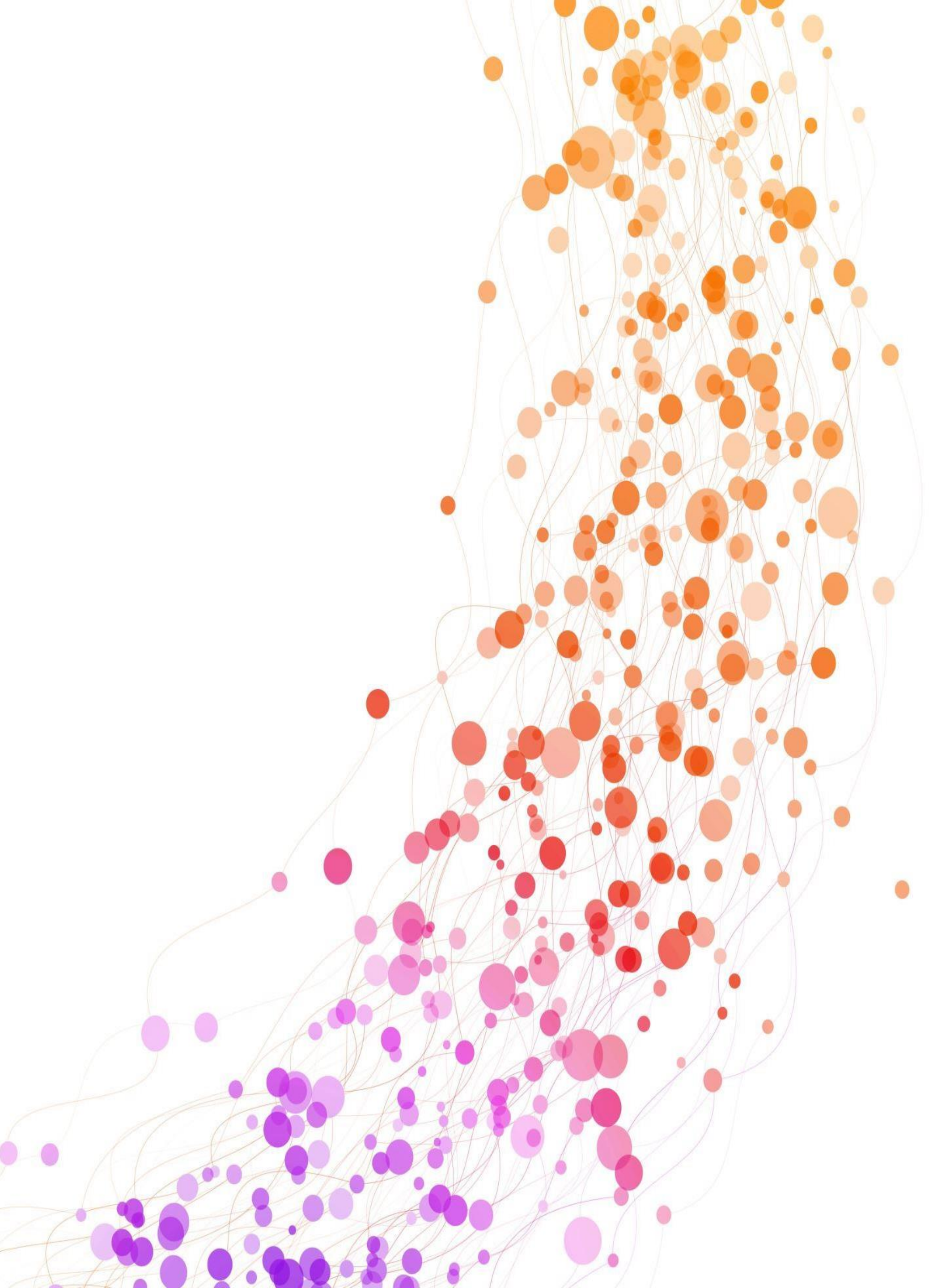


Recommender Systems

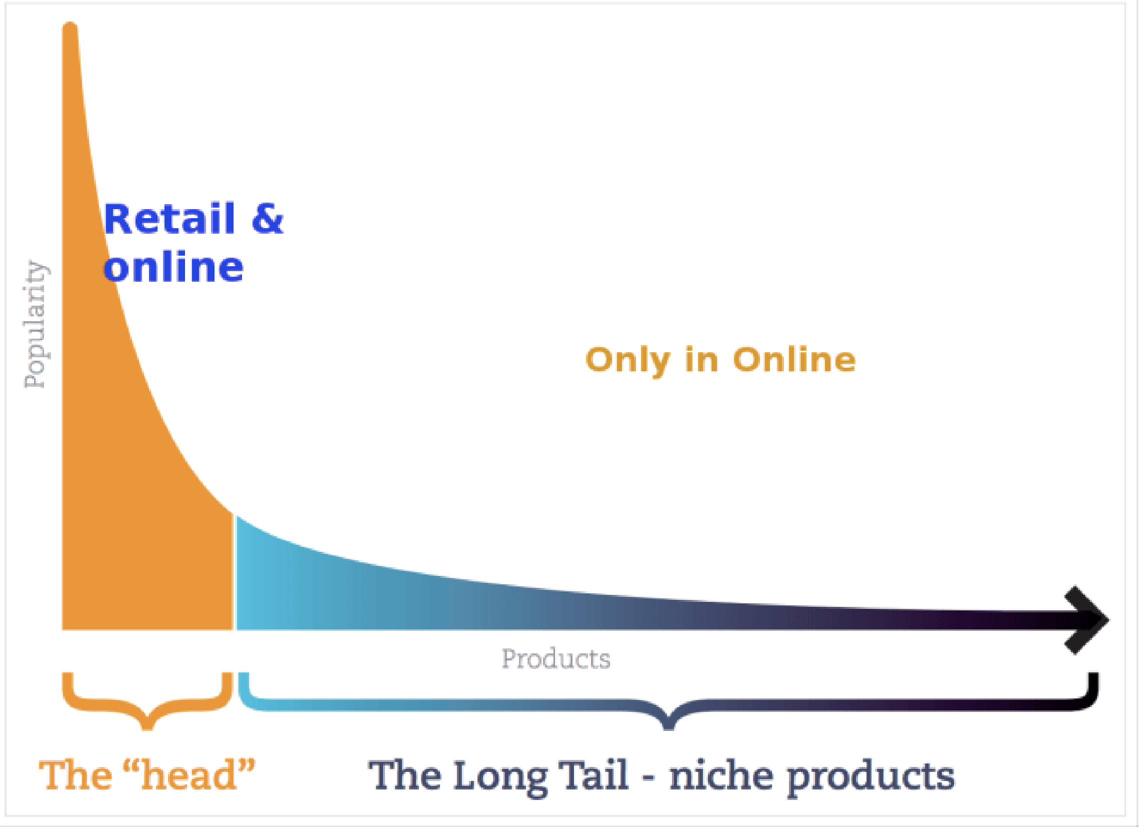
Individual Software Project

Supervisor: doc. RNDr. Iveta Mrázová, CSc.

Angie Granda

Introduction

- ❖ A **Recommendation System** is a subset of Artificial Intelligence that aims to learn patterns from data, which contains user-item interactions, and predicts how likely unknown items will be liked by the user
- ❖ ***Long Tail*** *Phenomenon*

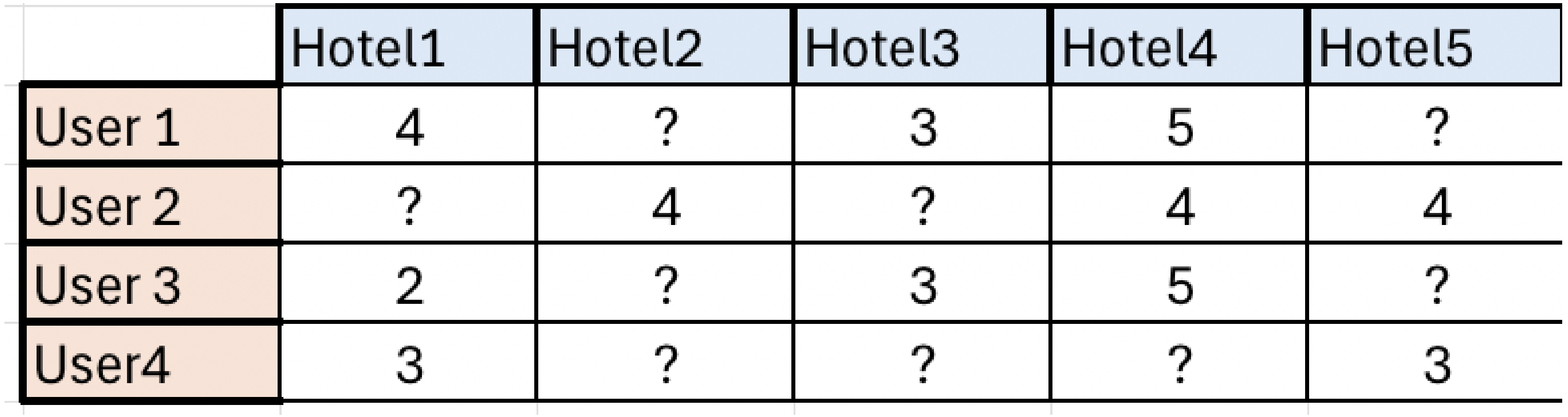


Recommender System Model

X = set of users, **S** = set of items, **R** = total ordering

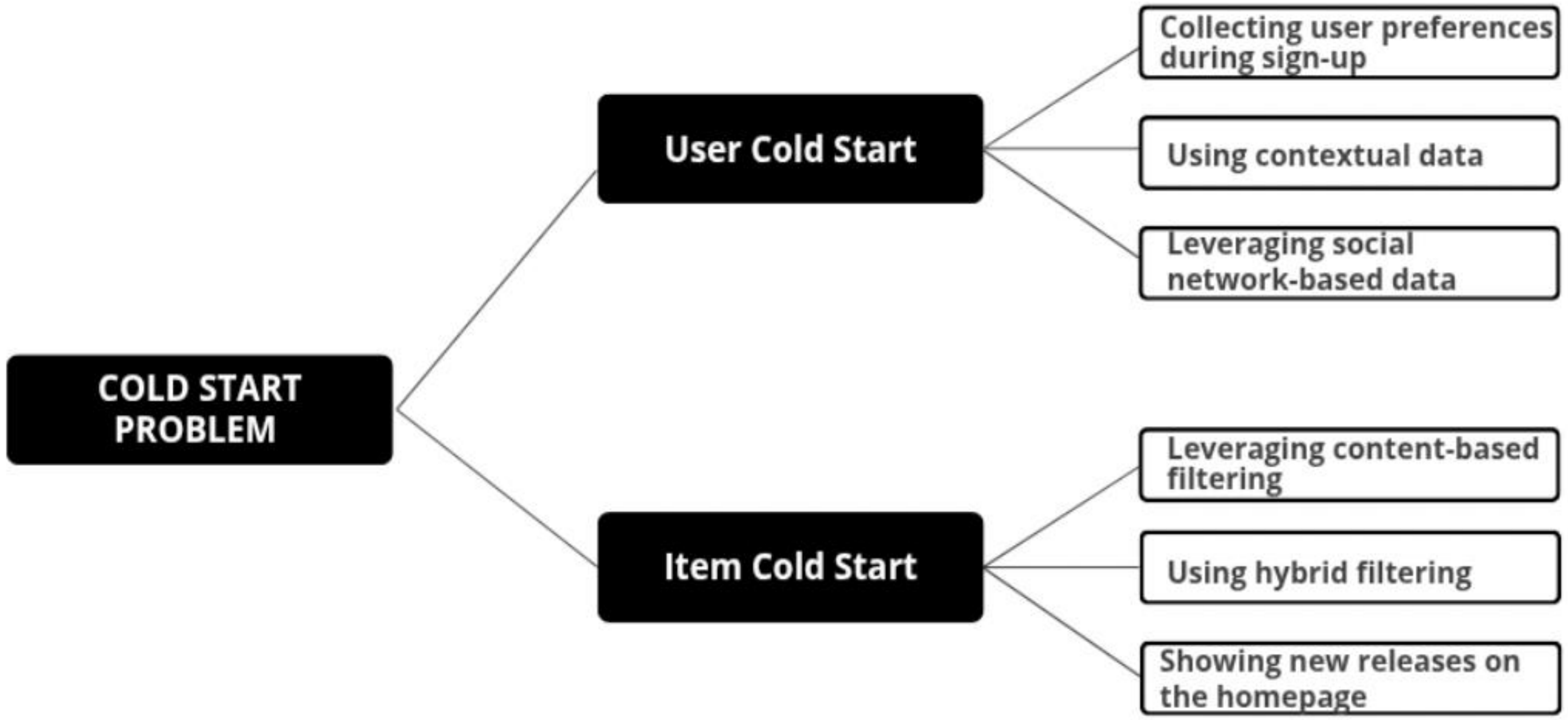
F: X * S -> R is the utility function, **F** fills the **utility matrix**.

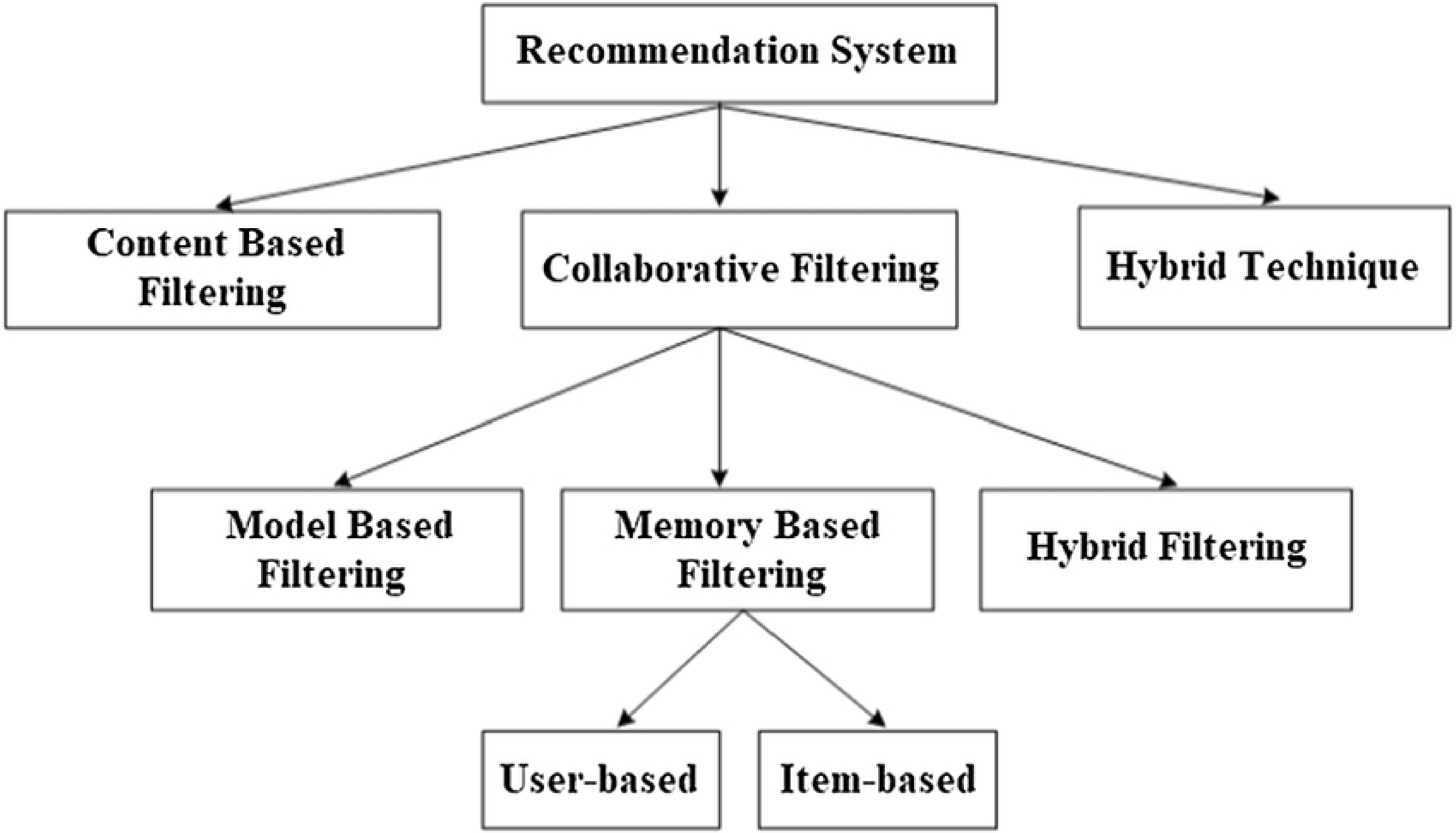
The **user-item value in the utility matrix** represent what is known about the degree of preference. Those values can be explicit or implicit.



	Hotel1	Hotel2	Hotel3	Hotel4	Hotel5
User 1	4	?	3	5	?
User 2	?	4	?	4	4
User 3	2	?	3	5	?
User4	3	?	?	?	3

The recommender system cannot accurately provide relevant suggestions to users when there is little or no historical information about them.


COLD START PROBLEM
User Cold Start
Collecting user preferences during sign-up
Using contextual data
Leveraging social network-based data
Item Cold Start
Leveraging content-based filtering
Using hybrid filtering
Showing new releases on the homepage


Recommendation System
Content Based Filtering
Collaborative Filtering
Hybrid Technique
Model Based Filtering
Memory Based Filtering
Hybrid Filtering
User-based
Item-based

Prediction heuristic - Similarity metrics

➢***Cosine Similarity:*** Focus on direction, not in magnitudes

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

➢***Pearson Correlation:***

It measures linear correlation between two vectors. It is scale-invariant.

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

➢**Jaccard Distance:**

Interest in shared presence of items only.

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

➢**Euclidean Distance:**

Measures actual differences in feature values. Sensitive to rating bias.

Content-Based R.S.

- Relevant characteristics of an item are represented by an **item profile**
- **User profile** summarizes the preferences of the user, weighted average of rated item profiles.

$$\text{UserProfile} = \frac{\sum_{i=1}^{N}(\text{rating}_i \cdot \text{feature\_vector}_i)}{\sum_{i=1}^{N}\text{rating}_i}$$

- **Prediction** is based on the similarity between user profile ***i*** and item profile ***j.***
- **Pros:** No need for data on other users, no item cold-start problem, able to recommend to users with unique tastes.
- **Cons:** Find appropriate features, user-cold start problem, overspecialization.

Collaborative Filtering. User-Based Model

Task: Consider user x. Find set N of other users whose ratings are like x's ratings. Estimate x's ratings based on ratings of users in N.

Mean-Center Overlapping-Items Cosine Similarity

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s \in S_{xy}} (\boldsymbol{r_{xs}} - \overline{\boldsymbol{r_x}})(\boldsymbol{r_{ys}} - \overline{\boldsymbol{r_y}})}{\sqrt{\sum_{s \in S_{xy}} (\boldsymbol{r_{xs}} - \overline{\boldsymbol{r_x}})^2} \sqrt{\sum_{s \in S_{xy}} (\boldsymbol{r_{ys}} - \overline{\boldsymbol{r_y}})^2}}$$

Prediction: Let Rx be the vector of user x's ratings. Let N be the set of users most similar to x who have rated item i.

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi} \quad r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}} \qquad r_{ij} = \bar{r}_i + \frac{\sum_k Similaries(u_i, u_k)(r_{kj} - \bar{r}_k)}{number\ of\ ratings}$$

Collaborative Filtering. Item-Based Model

Task: For user **x** and item **y**, find other N similar items to **y**, estimate rating for item **y** based on ratings of the similar items given by **x**

It can use same **similarity** metrics and **prediction** functions as in user-based model

Cons: If a user has rated few items, then most items can't be estimated

N(i;x) set of items similar to item i, rated by x.

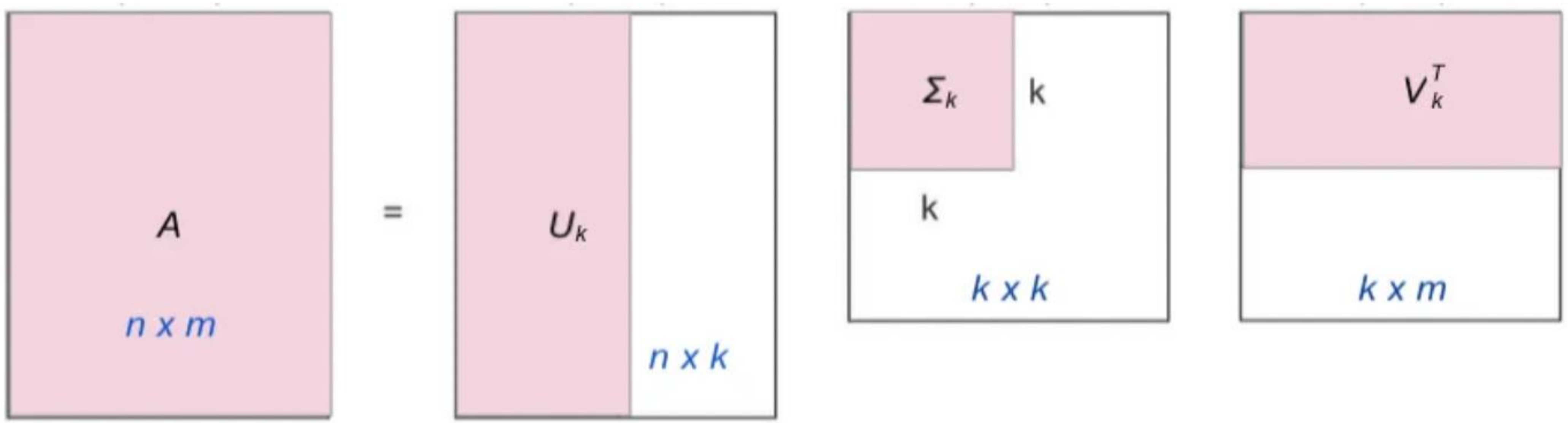
$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

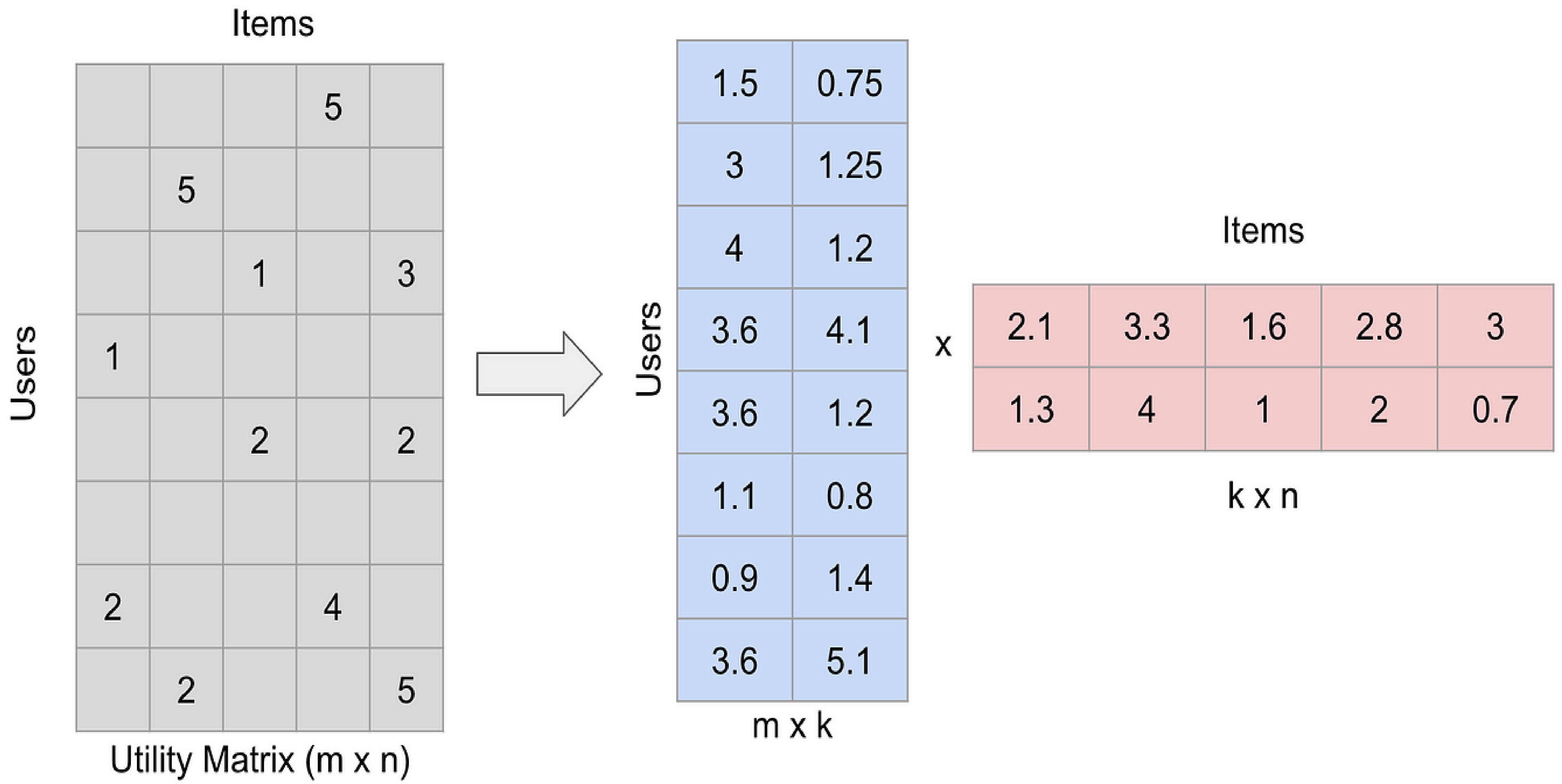
Collaborative Filtering - Matrix Factorization

Singular Value Decomposition (SVD) is a matrix factorization technique

$$A = U\Sigma V^T \qquad A \approx UV^T \text{ (when } \Sigma \text{ is absorbed into } U \text{ or } V\text{)}$$

Let ***pu*** ∈ U and ***qi*** ∈ V and rating of user ***u*** to item ***i*** be ***rui,*** then the prediction $\hat{\boldsymbol{r}}_{ui} = \boldsymbol{q}_i^T \boldsymbol{p}_u$




Items
Users
Utility Matrix (m x n)
Users
m x k
x
Items
k x n
1.5 0.75
3 1.25
4 1.2
3.6 4.1
3.6 1.2
1.1 0.8
0.9 1.4
3.6 5.1
2.1 3.3 1.6 2.8 3
1.3 4 1 2 0.7

Matrix Factorization - Funk MF

The loss function is **MSE**

$$\min_{q^\star, p^\star} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

usually with **L2 regularization** to prevent overfitting.

$$\min_{p,q,b_u,b_i} \sum (r_{ui} - (p_u^T q_i + \mu + b_u + b_i))^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

After taking partial derivatives with respect to vector **p** and **q** we obtain the following updating rules:

$$qi \leftarrow qi + \gamma \cdot (eui \cdot pu - \lambda \cdot qi)$$
$$pu \leftarrow pu + \gamma \cdot (eui \cdot qi - \lambda \cdot pu)$$

Mini-Batch Stochastic Gradient Descent

Matrix Factorization – ALS

Alternating Least Squares (**ALS**) uses the Sum Square Error Function

Algorithm: Fix V, solve for U using least squares. Then fix U, solve for V. Alternate until convergence.

Suppose that you fix V, then we want to solve: $\mathcal{L}_i(p_i) = \sum_{j \in \mathcal{I}_i} (r_{ij} - p_i^T q_j)^2 + \lambda \|p_i\|^2$

$Q_i \in \mathbb{R}^{|\mathcal{I}_i| \times k}$: matrix whose rows are the q_j^T vectors for all $j \in \mathcal{I}_i$

$r_i \in \mathbb{R}^{|\mathcal{I}_i|}$: vector of the known ratings r_{ij}

$$\mathcal{L}_i(p_i) = \|r_i - Q_i p_i\|^2 + \lambda \|p_i\|^2$$

$$\nabla_{p_i} \mathcal{L}_i = -2Q_i^T (r_i - Q_i p_i) + 2\lambda p_i$$

$$-2Q_i^T (r_i - Q_i p_i) + 2\lambda p_i = 0$$

$$Q_i^T Q_i p_i + \lambda p_i = Q_i^T r_i$$

$$p_i = \left(\sum_{j \in \mathcal{I}_i} q_j q_j^T + \lambda I \right)^{-1} \sum_{j \in \mathcal{I}_i} r_{ij} q_j$$

$$q_j = \left(\sum_{i \in \mathcal{U}_j} p_i p_i^T + \lambda I \right)^{-1} \sum_{i \in \mathcal{U}_j} r_{ij} p_i$$

Hybrid Recommender Systems

- **Parallel use of several systems**

Multiple recommender systems operate independently and simultaneously on the same input. Their outputs are then combined.

- **Monolithic exploiting different features**

A single recommendation system that uses a wide variety of features — e.g., user profiles, item metadata, past interactions, contextual signals

- **Pipelined invocation of different systems**

Multiple recommenders are arranged in sequence, where the output of one system feeds into the next.

Data Split and Evaluation

Techniques to split the data:

- Leave One Last
- Temporal user-based
- Temporal Global

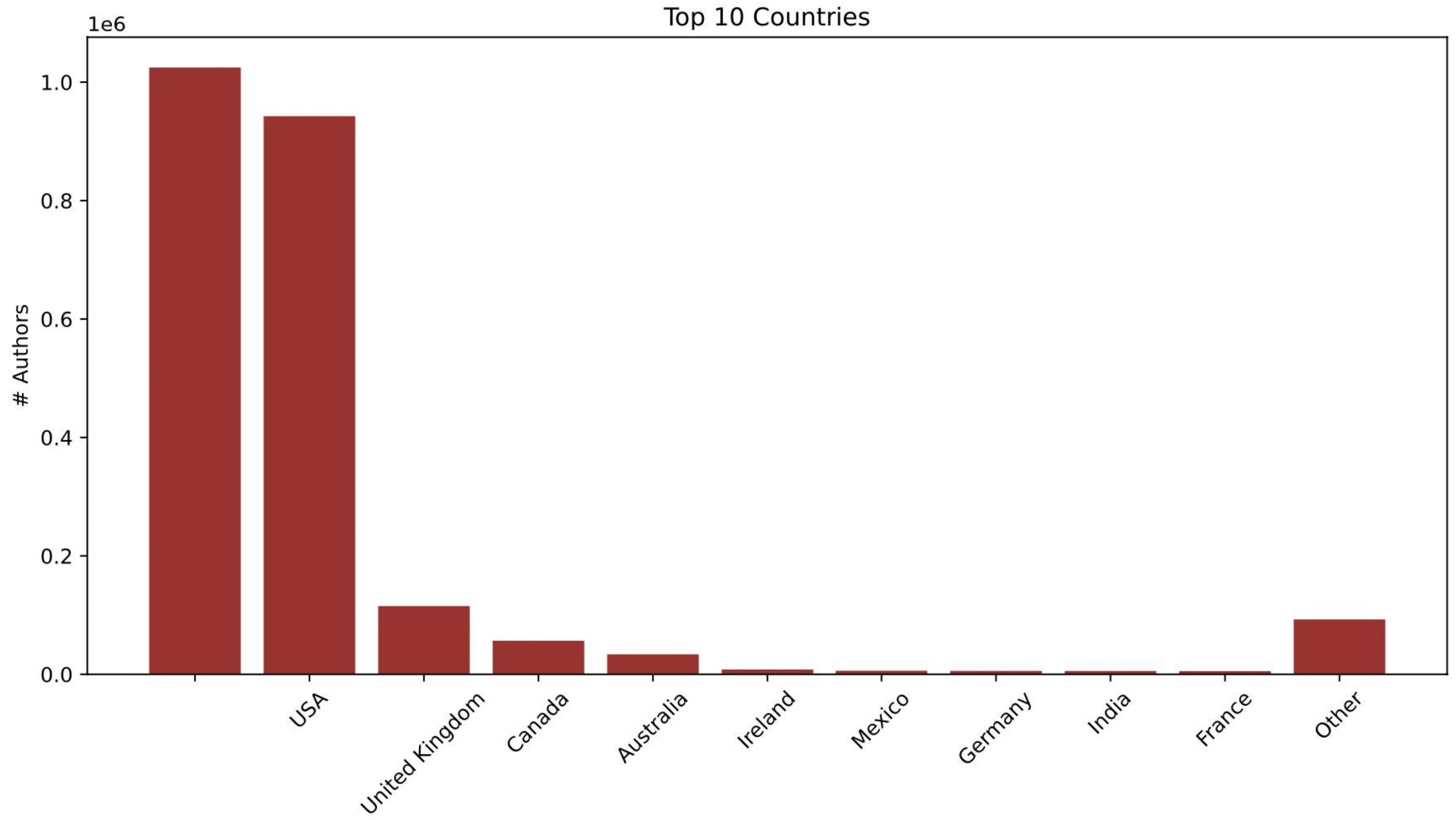
Evaluating predictions metrics:

- MAE, MSE, RMSE

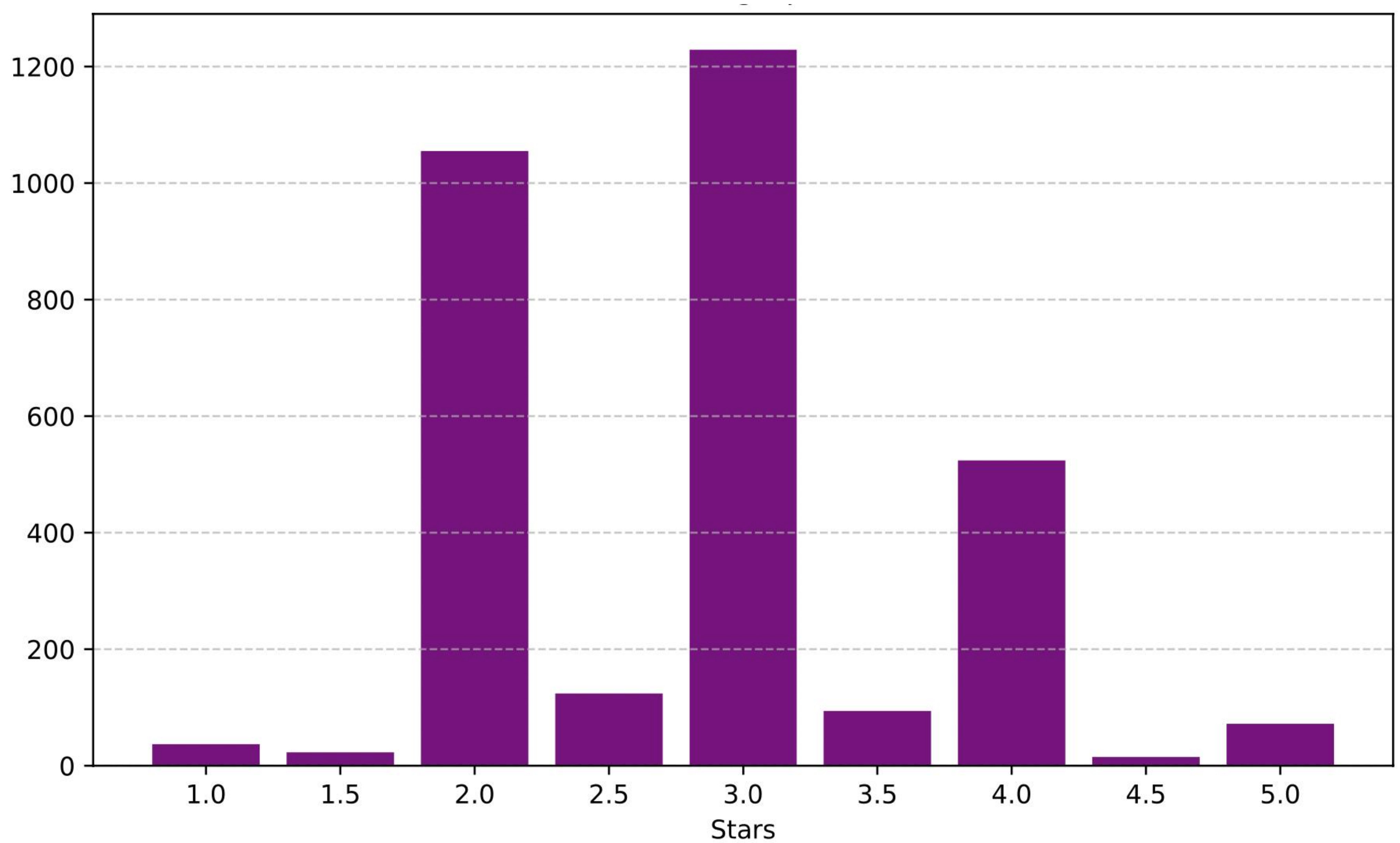
Evaluating lists of recommendation (based on relevancy levels):

- Precision, Mean Average Precision
- Recall

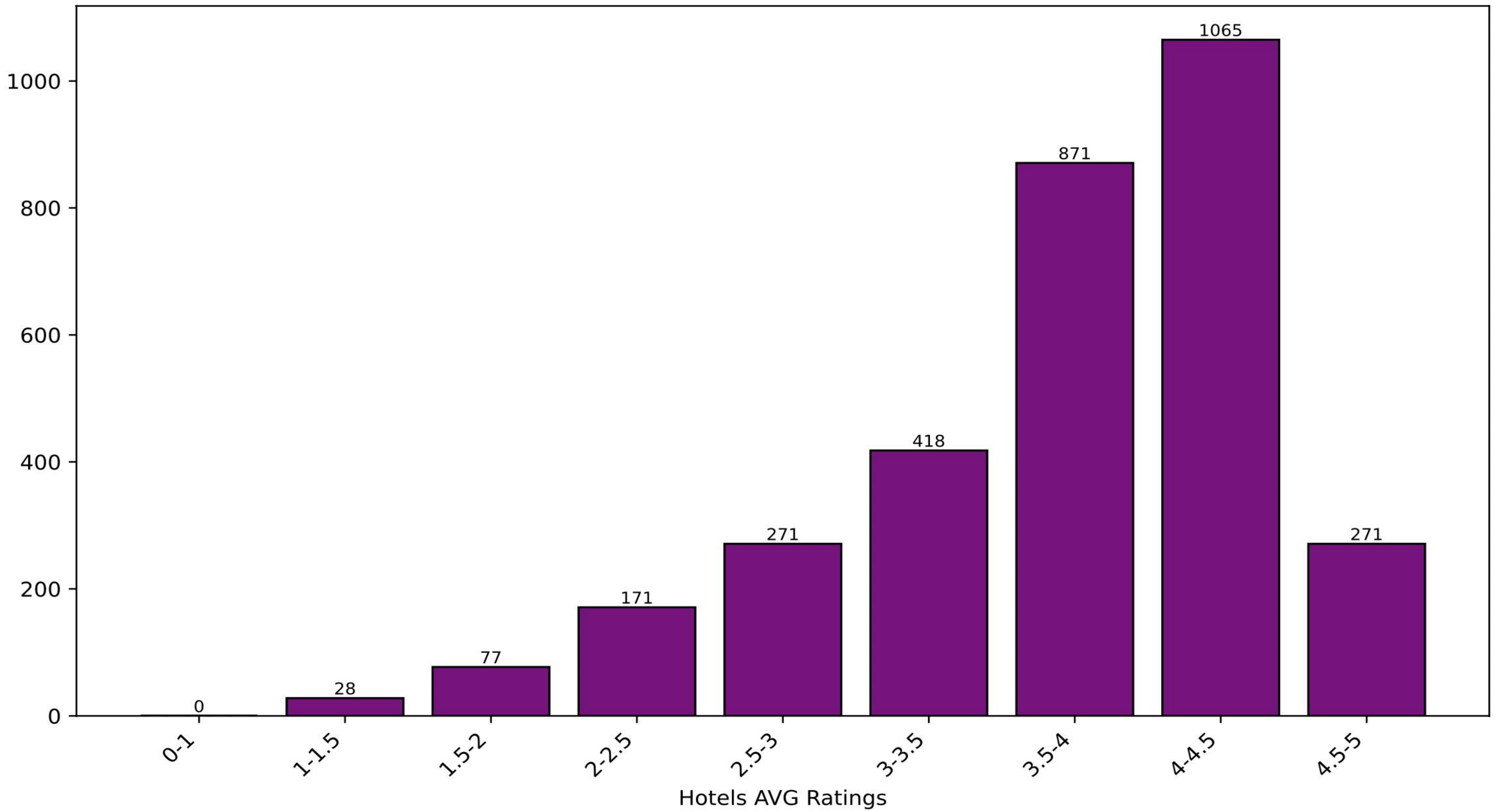
Dataset - Hotels



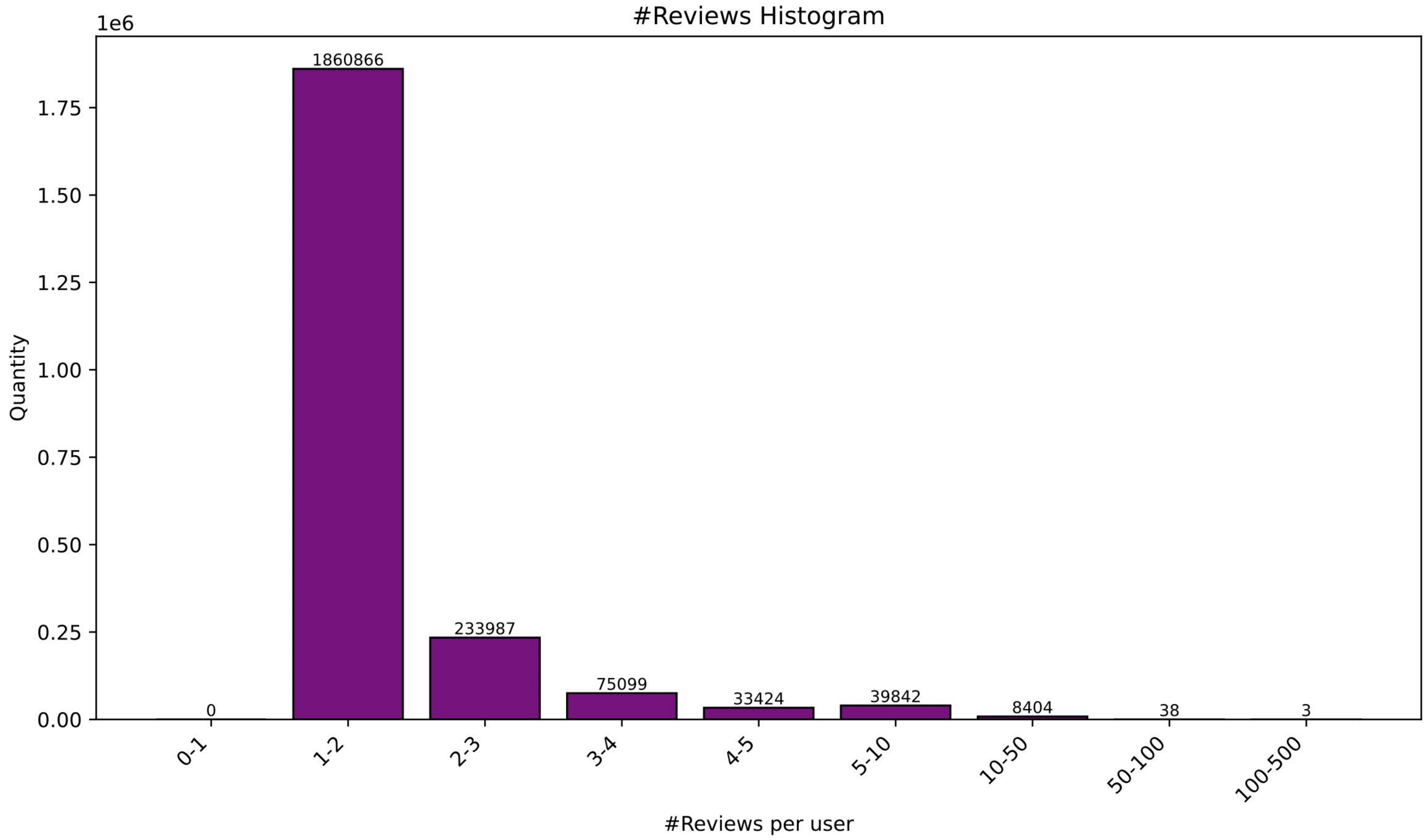
Dataset - Hotels



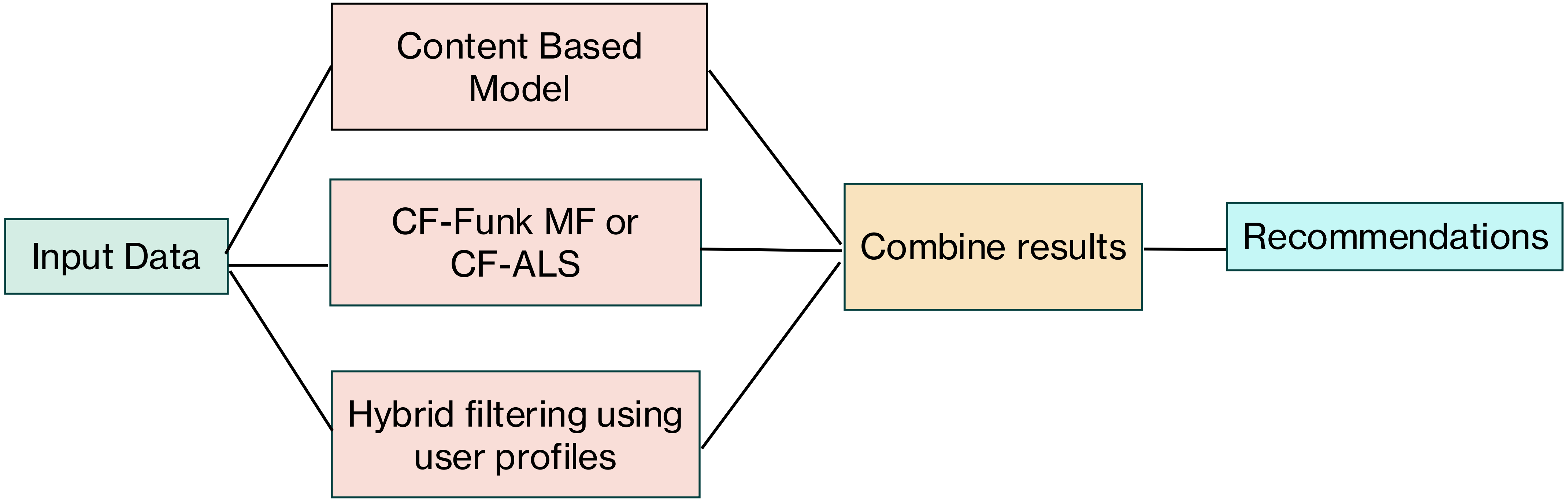
Dataset - Hotels



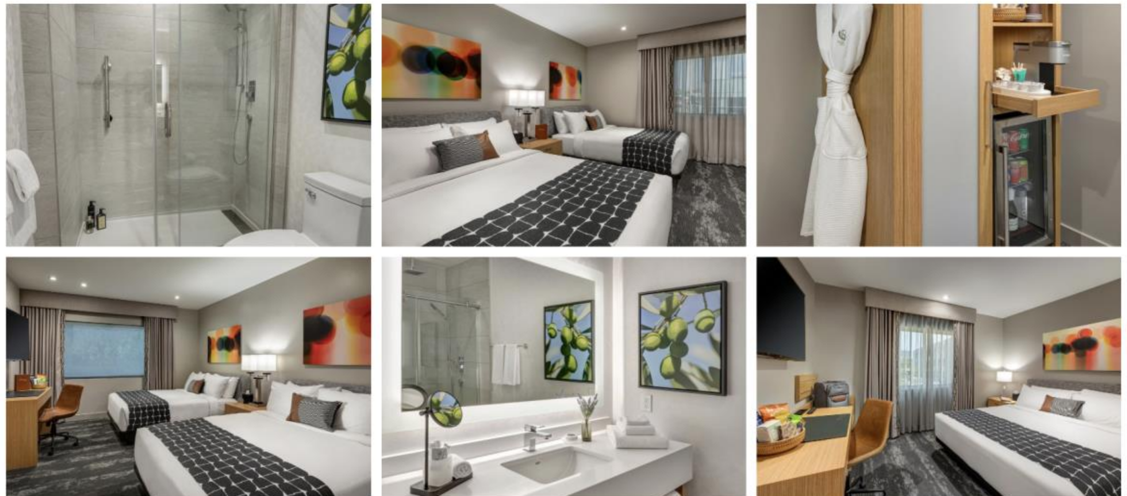
Dataset - Authors



Project Model



User rating hotels 3-4 overall and with a budget of 150-500$



Remaining Work

- Finalize and evaluate each model throughtly
- Test hybrid filtering by integrating user-based KNN using user profiles.
- Assess regional influence on user preferences:
  Given users with a substantial number of ratings, we'll evaluate how effectively their preferences can be predicted using region-based average user profiles. Are regional-based predictions actually meaningful?

Thank you for your attention!

Questions?