Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model

Matrix Factorization Techniques For Recommender Systems

2022. 01. 04. Seung Yoon Choi

Contents

01 Abstract

“Advanced Collaborative Filtering Model” & “New Evaluation Metric”

- Past transactions are analyzed in order to establish connections between users and products.
- Two successful approaches
  - Latent factor models : directly profile both users and products
  - Neighborhood models : analyze similarities between products or users
- Some innovations to both approaches.
- Two models were smoothly merged, building a more accurate combined model.
- New evaluation metric based on their performance at a top-K recommendation task.

“ Contents-based Filtering(CBF) Approach”

- Creates a profile for each user or product to characterize its nature.

- Content-based strategies require gathering external information that might not be available or easy to collect.

“ Collaborative Filtering(CF) Approach”

Neighborhood Model

- Using data from a small number of users with similar item consumption tendencies.

- Advantage of detail recommendations.
- Disadvantage of using only some data.

Latent Factor Models

- Transforming both items and users to the same latent factor space.

- Advantage of using the entire data.
- Disadvantage of detail recommendations.

○ Past transaction ○ Cold Start

“ Integrating Different Forms of User Input into Models!!” (Netflix Prize Competition)

Explicit Data

- The rating given by the user for the content.
- Cannot always be used…

Implicit Data

- Indirectly reflect opinion through obs erving user behavior.
- Ex) Purchase history, browsing history, search patterns, mouse movements, and so on.
- Abundant amount relative to explicit data.

03 Neighborhood Model

“ Baseline Model”

$$b_{ui} = \mu + b_u + b_i$$

μ : overall average rating
b_u : observed deviations of user u
b_i : observed deviations of item i

$$\text{Loss Function} = \min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \underbrace{(\sum_u b_u^2 + \sum_i b_i^2)}_{\text{regularizing term}}$$

03 Neighborhood Model

“ Item-based Model”

Central to item-oriented approaches is a similarity measure between items.

$$s_{ij} \overset{\text{def}}{=} \frac{n_{ij}}{n_{ij} + \lambda_2} \rho_{ij}$$

- ρ_{ij} : Pearson correlation coefficient
- n_{ij} : the number of users that rated both i and j
- λ_2 : typical value is 100

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}$$

- $S^k(i;u)$: k items rated by u, which are most similar to i.

03 Neighborhood Model

"Global Weights independent to specific user"

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij}$$

w_{ij} : not user specific
$R(u)$: all the items for which ratings by u are available

03 Neighborhood Model

“Implicit Feedback”

$$\hat{r}_{ui} = \mu + b_u + b_i + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} + \sum_{j \in \mathrm{N}(u)} c_{ij}$$

$N(u)$: all the items for which u provided an implicit preference
c_{ij} : offsets added to baseline estimates

03 Neighborhood Model

"Scaling"

- Previous model somewhat overemphasizes the dichotomy between heavy raters and those that rarely rate.

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} c_{ij}$$

03 Neighborhood Model

"Final : Reducing Target"

- In previous model, it is expected that the most influential weights will be associated with items similar to i.

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij}$$

$$\mathrm{R}^k(i;u) \stackrel{\text{def}}{=} \mathrm{R}(u) \cap \mathrm{S}^k(i)$$

$$\mathrm{N}^k(i;u) \stackrel{\text{def}}{=} \mathrm{N}(u) \cap \mathrm{S}^k(i)$$

Loss Function =

$$\min_{b_*, w_*, c_*} \sum_{(u,i) \in \mathcal{K}} \left(r_{ui} - \mu - b_u - b_i - |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij} - |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} \right)^2$$

$$+ \lambda_4 \left(b_u^2 + b_i^2 + \sum_{j \in \mathrm{R}^k(i;u)} w_{ij}^2 + \sum_{j \in \mathrm{N}^k(i;u)} c_{ij}^2 \right)$$

03 Neighborhood Model

“Stochastic Gradient Descent ”

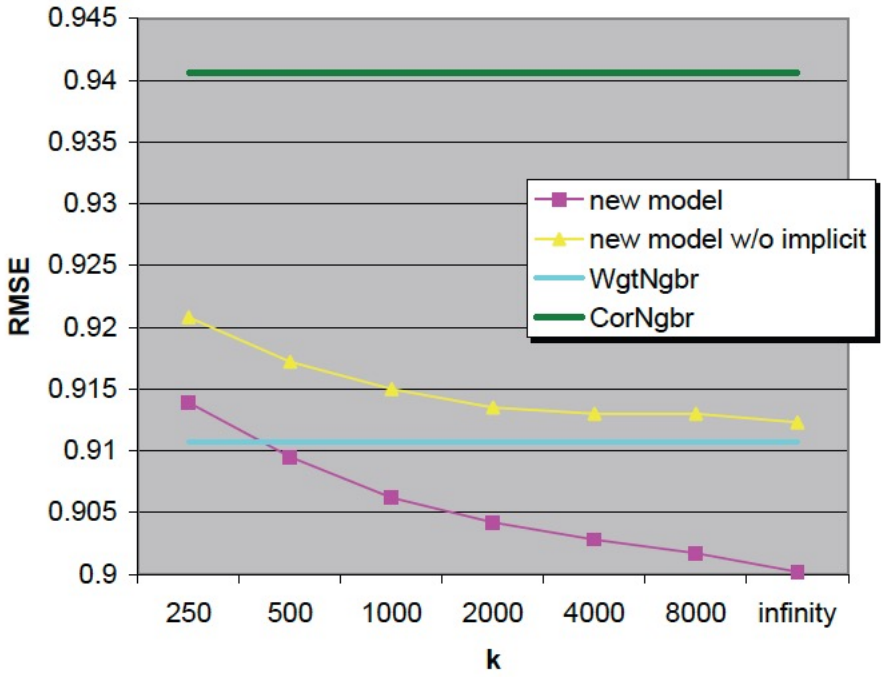
Modify the parameters by moving in the opposite direction of the gradient.

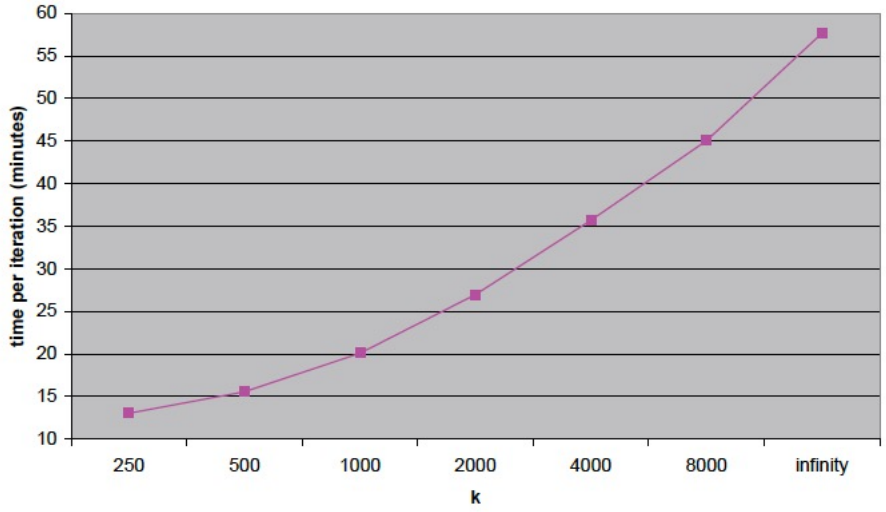
- $b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda_4 \cdot b_u)$
- $b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda_4 \cdot b_i)$
- $\forall j \in \mathrm{R}^k(i;u)$:
  $w_{ij} \leftarrow w_{ij} + \gamma \cdot \left(|\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_4 \cdot w_{ij} \right)$
- $\forall j \in \mathrm{N}^k(i;u)$:
  $c_{ij} \leftarrow c_{ij} + \gamma \cdot \left(|\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_4 \cdot c_{ij} \right)$

- γ and λ_4 : determined by cross-validation (we use γ=0.005, $\lambda_4 = 0.002$ for the Netflix data)
- Increasing k always benefits the accuracy of the results on the test set.
- The choice of k reflect a tradeoff between prediction accuracy and computational cost.

03 Neighborhood Model

“Performance”





"Latent Factor Model"

Latent factor models are

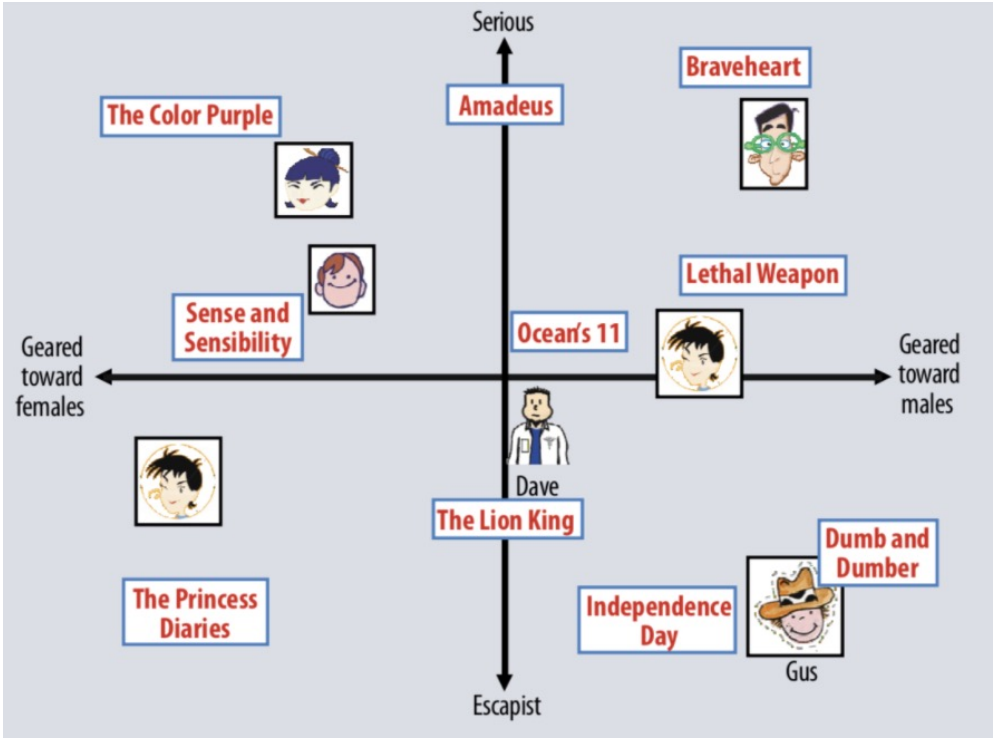
an approach that tries to explain the ratings

by characterizing both items and users on 20 to 100 factors

inferred from the ratings patterns.

$$\hat{r}_{ui} = q_i^T p_u$$

Different types of input data



q_i : measure how many positive or negative factors an item has.
p_u : measure how much positive or negative has preference for high items.

“Latent Factor Model”

- To apply SVD to the CF domain, a user-item rating matrix must be factored.
- This often leads to difficulties due to the high percentage of missing values due to the sparseness of the user-item rating matrix.
- Previous System
  - Focus on imputation
  - Cost increasing
  - Distortion
- Recent Studies
  - Only the observed grades are directly modeled.

"Learning Algorithm(1) - SGD"

1. For each given training case, the system predicts r_{ui} and computes the associated prediction error.

$$e_{ui} \overset{def}{=} r_{ui} - q_i^T p_u$$

2. Modifying the parameters by a magnitude proportional to γ in the opposite direction of the gradient.

- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$
- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$

- Easy to implement and relatively fast!
- In some cases, ALS benefits..

"Learning Algorithm(2) - ALS"

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

- q_i & p_u : unknown -----> Alternatively fix them.
- Proceed until each stage converges.

When ALS is favorable?

- System can use parallelization
  - This gives rise to potentially massive parallelization of the algorithm

- Systems centered on implicit data
  - Training set cannot be considered sparse
  - Looping over each single training case would not be practical

04 Latent Factor Models

“Adding Biases”

$$\hat{r}_{ui} = \underbrace{\mu}_{1} + \underbrace{b_i}_{2} + \underbrace{b_u}_{3} + \underbrace{q_i^T p_u}_{4}$$

1. Global Average
2. Item Bias
3. User Bias
4. User-Item Interaction

"Paterek's Model"

$$\hat{r}_{ui} = b_{ui} + q_i^T[(\Sigma_{(j \in R(u))} x_j)/\sqrt{|R(u)|}]$$

- User's formula for latent vector has been modified.
- x_j : item's latent
- User u's latent vector is modeled as the sum of item j's latent vector consumed by user

"Asymmetric SVD"

$$\hat{r}_{ui} = b_{ui} + q_i^T [|R(u)|^{-\frac{1}{2}} \Sigma_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{-\frac{1}{2}} \Sigma_{j \in N(u)} y_j]$$

- Calculate the user's latent vector by using the explicit & implicit data.

04 Latent Factor Models

“SVD++”

$$\hat{r}_{ui} = b_{ui} + q_i^T(p_u + |N(u)|^{-\frac{1}{2}}\Sigma_{j\in N(u)}y_j)$$

- The model that reports the best performance of the Latent Factor Model in this paper.

04 Latent Factor Models

"Performance"

Model	50 factors	100 factors	200 factors
SVD	0.9046	0.9025	0.9009
Asymmetric-SVD	0.9037	0.9013	0.9000
SVD++	0.8952	0.8924	0.8911

Table 1: Comparison of SVD-based models: prediction accuracy is measured by RMSE on the Netflix test set for varying number of factors (f). Asymmetric-SVD offers practical advantages over the known SVD model, while slightly improving accuracy. Best accuracy is achieved by SVD++, which directly incorporates implicit feedback into the SVD model.

"Additional Advanced Models"

Temporal Dynamics

$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

Confidence Levels

$$\min_{p^*, q^*, b^*} \sum_{(u,i)\in\kappa} c_{ui}(r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

An Integrated Model

$$\hat{r}_{ui} = b_{ui} + |R^k(i;u)|^{-\frac{1}{2}} \Sigma_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \Sigma_{j \in N^k(i;u)} c_{ij} + q_i^T (p_u + |N(u)|^{-\frac{1}{2}} \Sigma_{j \in N(u)} y_j)$$

Optimization

- $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$
- $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$
- $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j) - \lambda_7 \cdot q_i)$
- $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$
- $\forall j \in \mathrm{N}(u)$:
  $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |\mathrm{N}(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_7 \cdot y_j)$
- $\forall j \in \mathrm{R}^k(i;u)$:
  $w_{ij} \leftarrow w_{ij} + \gamma_3 \cdot \left(|\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_8 \cdot w_{ij} \right)$
- $\forall j \in \mathrm{N}^k(i;u)$:
  $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot \left(|\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_8 \cdot c_{ij} \right)$

Performance

	50 factors	100 factors	200 factors
RMSE	0.8877	0.8870	0.8868
time/iteration	17min	20min	25min

Baseline : 0.9514

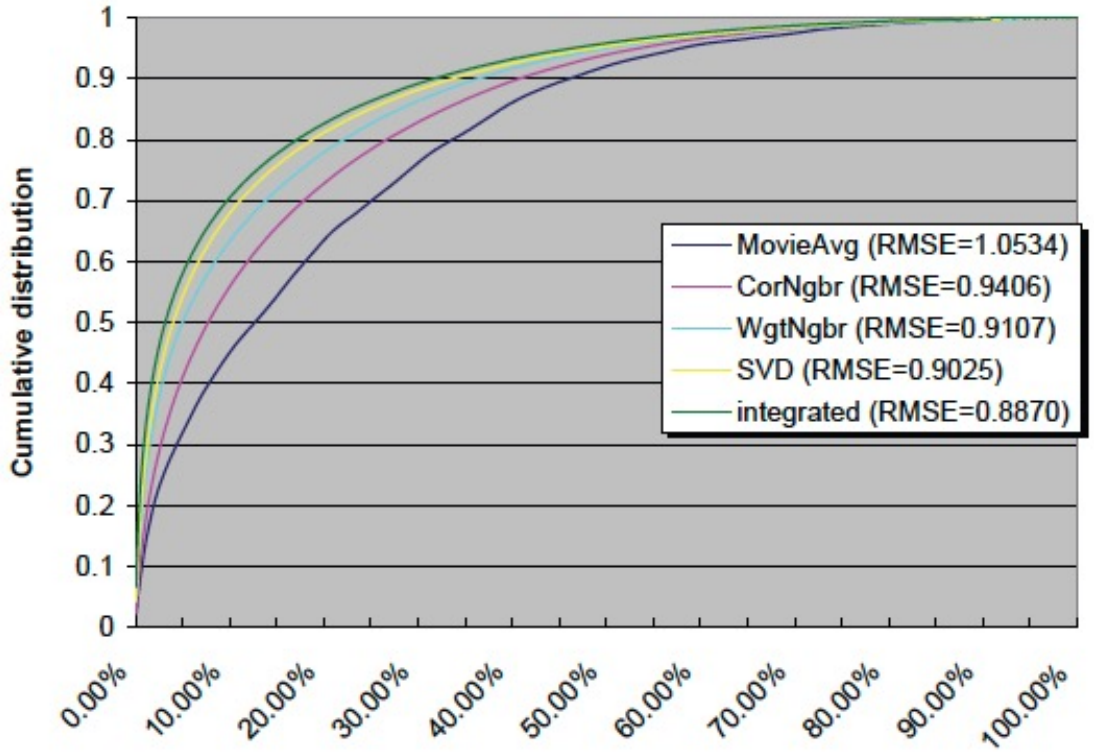
06 New Evaluation Metric

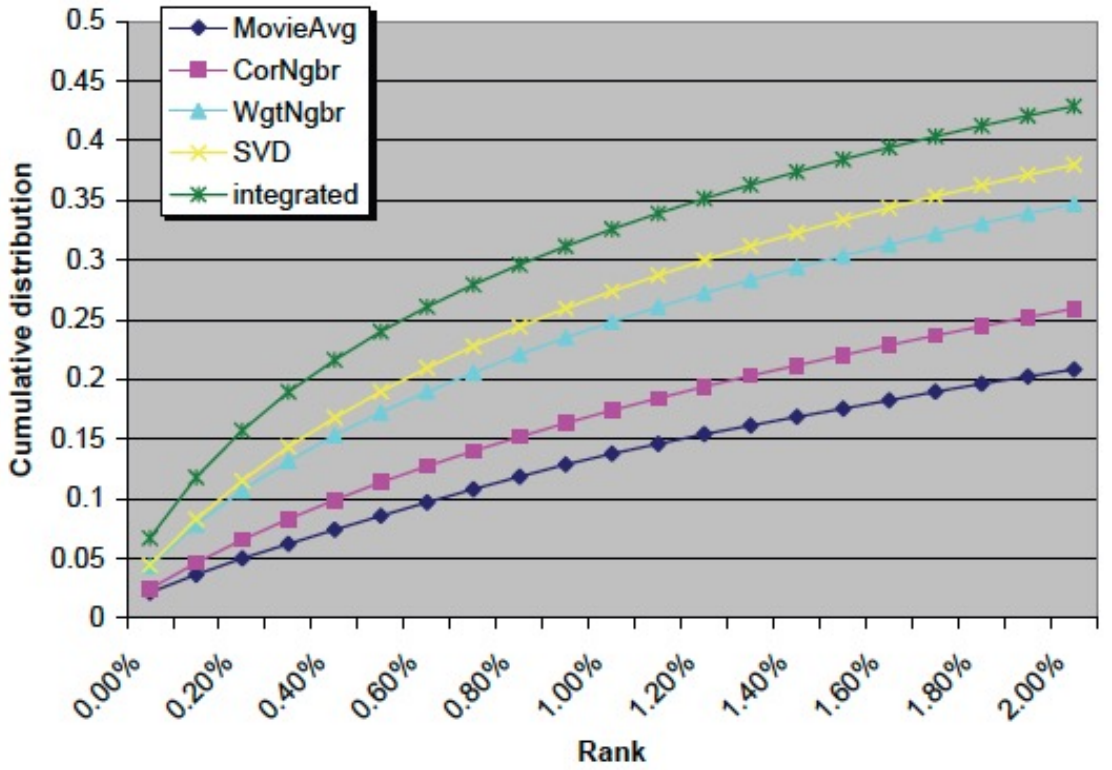
“Top-K Evaluation”

- Goal is to find the relative place of ”interesting movies” within the total order of movies sorted by predicted ratings for a specific user.
- For each such movie i, rated 5-stars by user u, select 1000 additional random movies a nd predict the ratings by u for i and for the other 1000 movies.
- Order the 1001 movies based on their predicted rating, in a decreasing order.

06 New Evaluation Metric

“Top-K Evaluation”





Discussion

1 **Neighborhood Model**

- Global optimization
- Implicit feedback

2 **Latent Factor Model**

- Implicit feedback

3 **Integrated Model**

- Neighborhood + SVD++

4 **New Evaluation Metric**

Top-K Recommender

Thank You