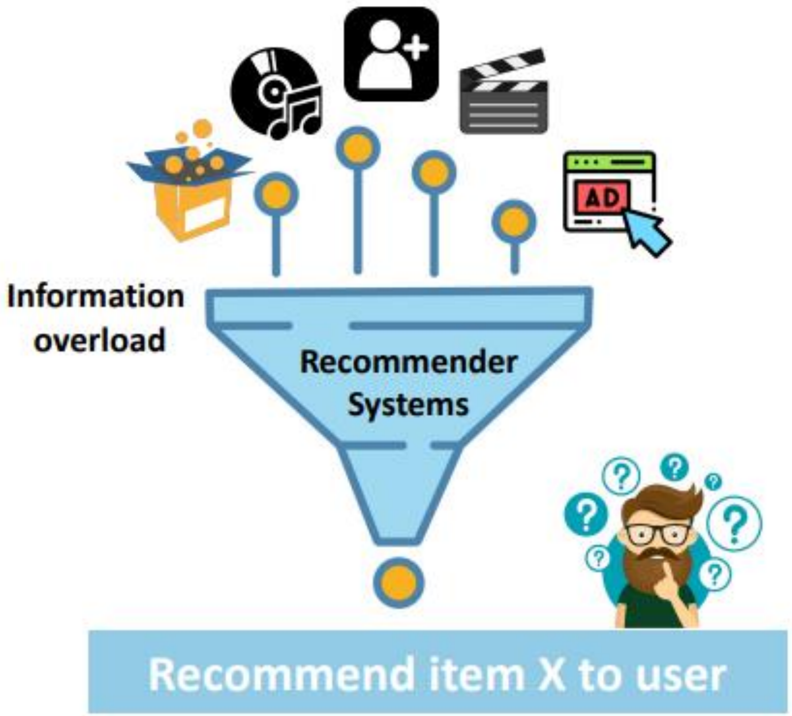
Recommender Systems

http://cleverdata.io/basket-analysis-machine-learning/

The Age of Search has come to an end
…long live the Age of Recommendation!
Chris Anderson "The Long Tail"

Recommender systems



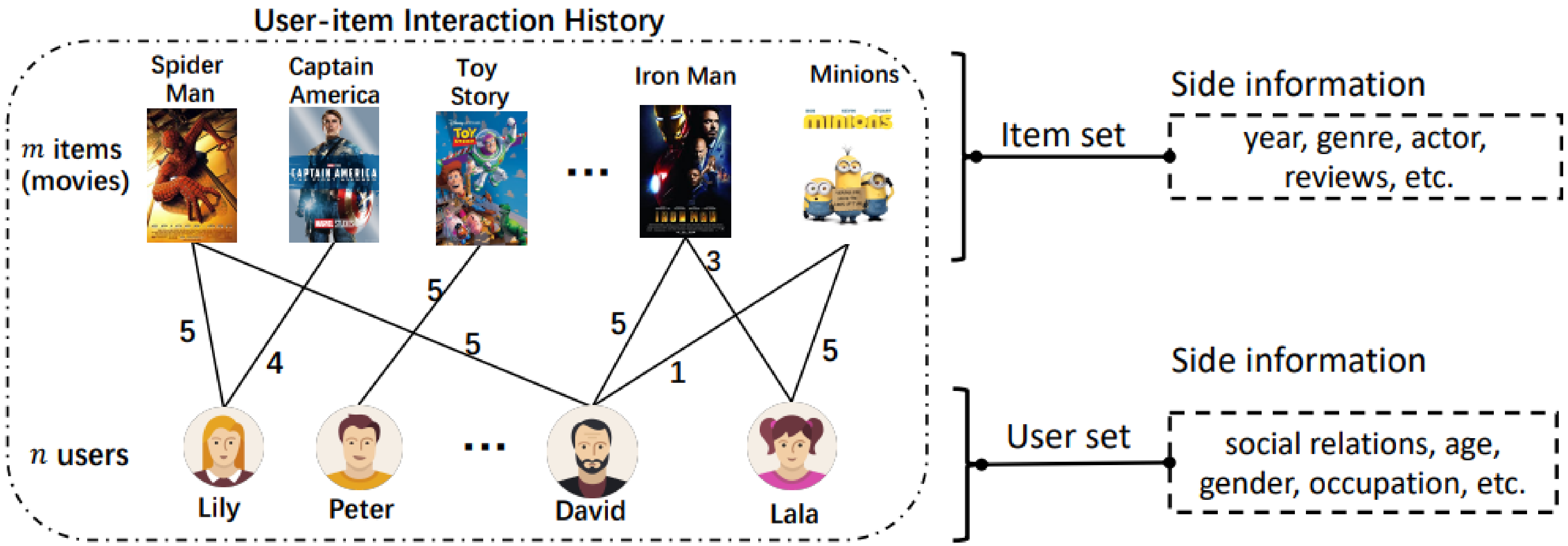
Items can be: Products, News, Movies, Videos, Friends, etc.

Personalized Web-based applications that provide users with personalized recommendations about content they may be interested in

Problem formulation

INPUT

Historical user-item interactions or additional side information (e.g., social relations, item's knowledge, etc.)

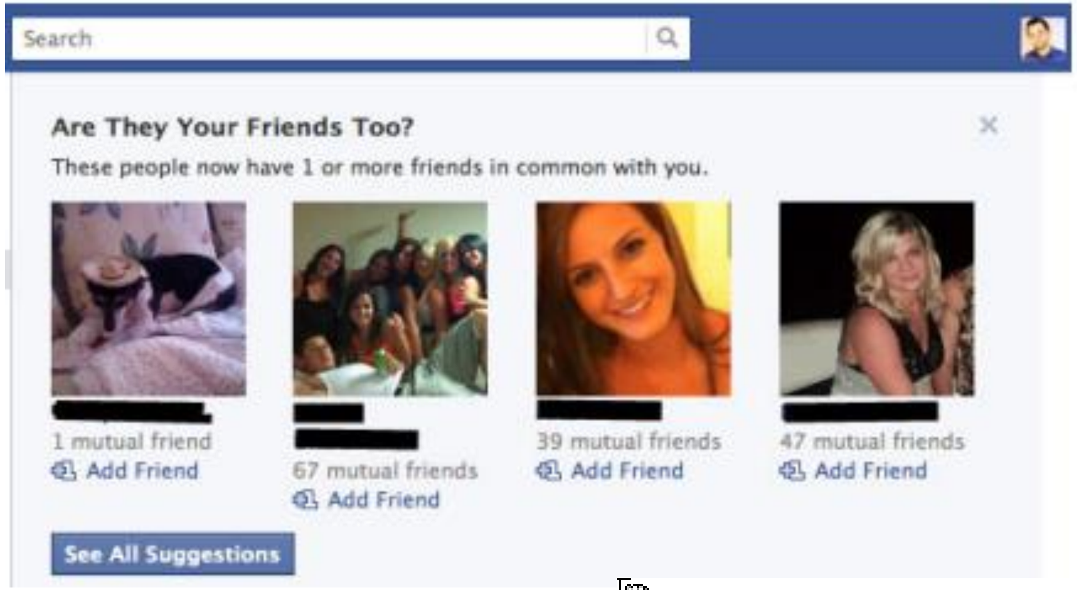


Tutorial website: https://deeprs-tutorial.github.io

Problem formulation

OUTPUT

Predict how likely a user would interact with a target Item (e.g., click, view, or purchase)



Tutorial website: https://deeprs-tutorial.github.io

Applications

Application: E-commerce,

- Recommendation has been widely applied in online services

amazon

ebay

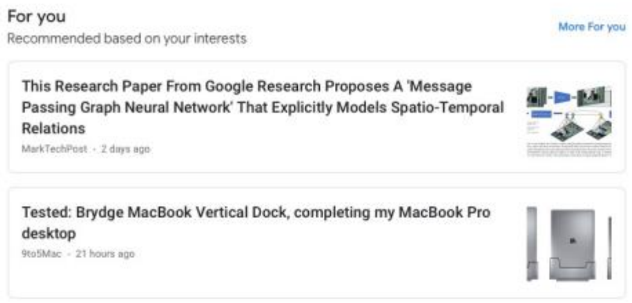
淘宝网
Taobao.com

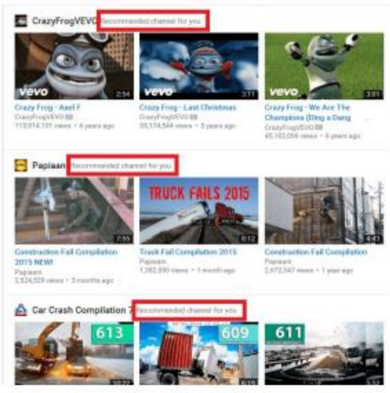
JD.COM

Application: Content sharing

- Recommendation has been widely applied for content sharing

News/Video/Image Recommendation





Application: Social

- Recommendation has been widely applied in social networks

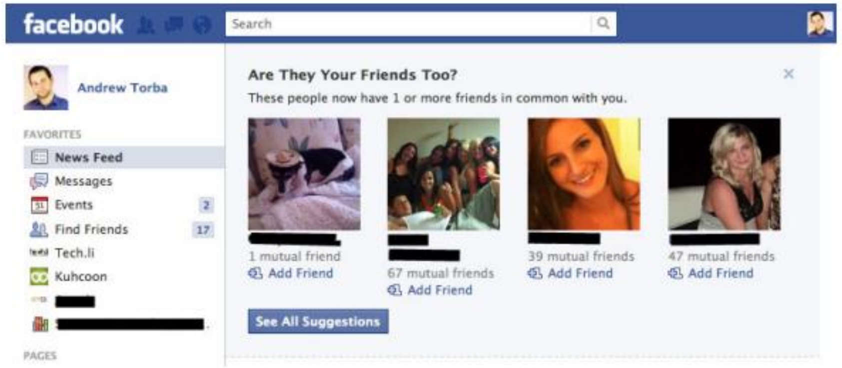
amazon

ebay

淘宝网
Taobao.com

JD.COM

Friend Recommendation



The value of recommendations

Source: McKinsey

The value of recommendations: Amazon

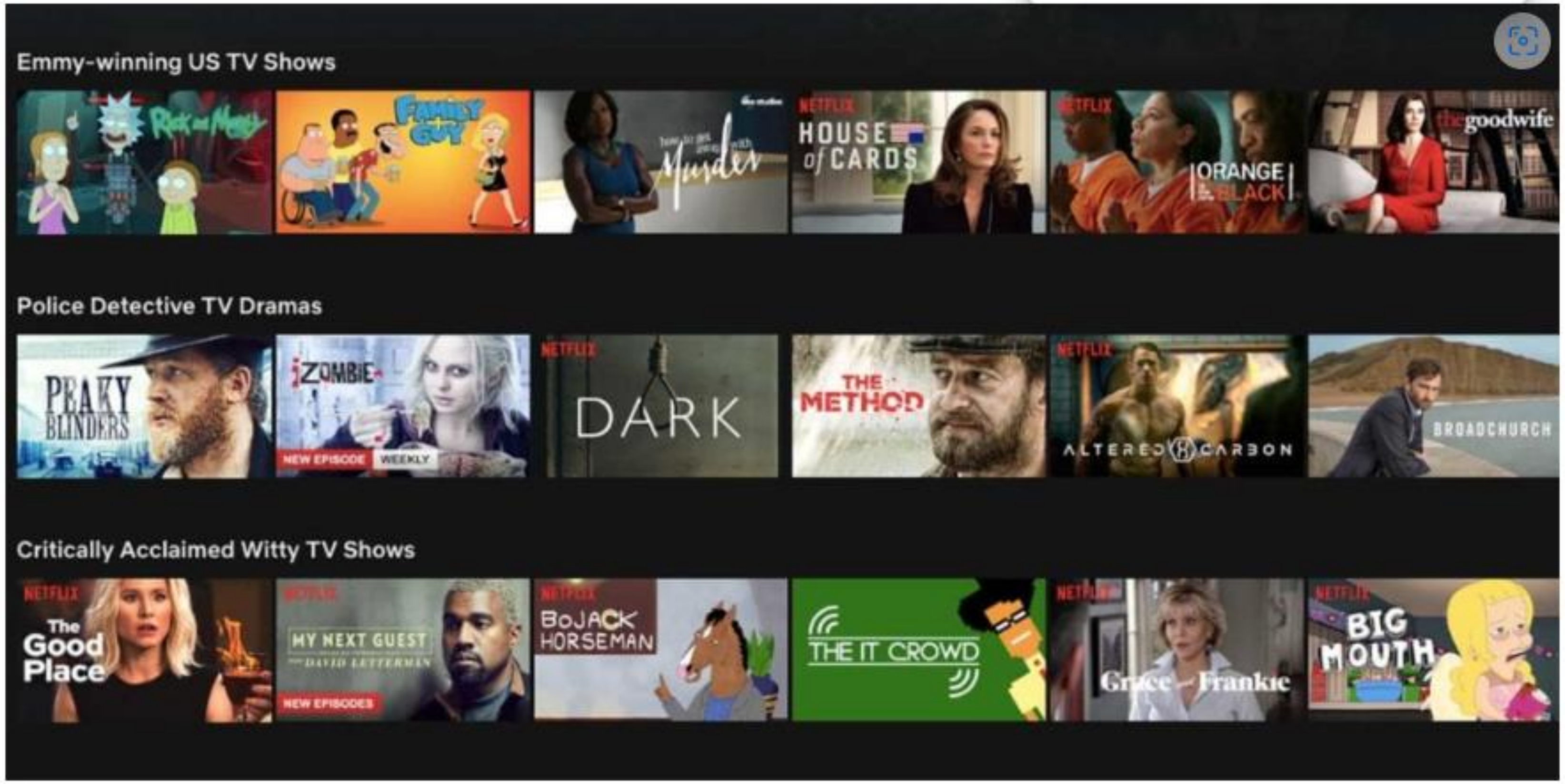
- [According to McKinsey,](#) 35% of Amazon purchases are thanks to recommendation systems.

[Sistema di suggerimento – Amazon Personalize – Amazon Web Services](#)

The value of recommendations: Netfix

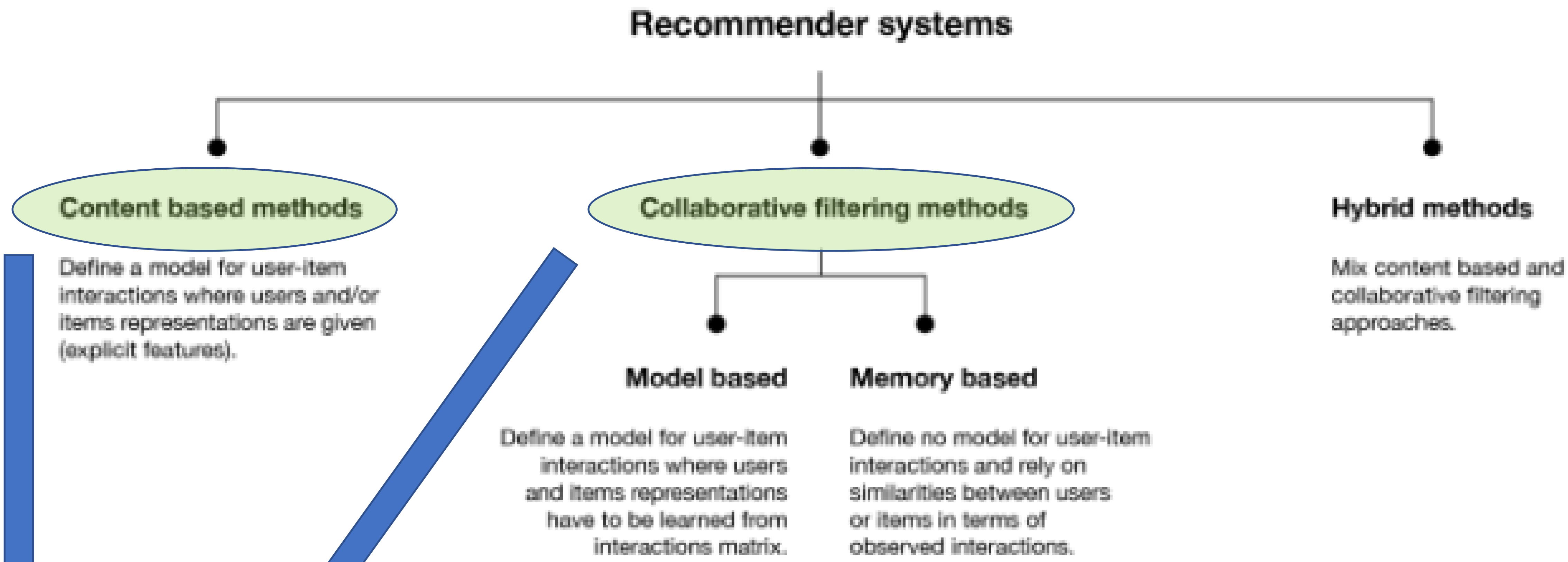
- Mckinsey study we mentioned above highlights that 75% of Netflix viewing is driven by recommendations. In fact, Netflix is so obsessed with providing the best results for users that they held data science competitions called Netflix Prize where one with the most accurate movie recommendation algorithm wins a prize worth $1,000,000.

- How Netflix's Recommendations System Works



Taxonomy

Taxonomy



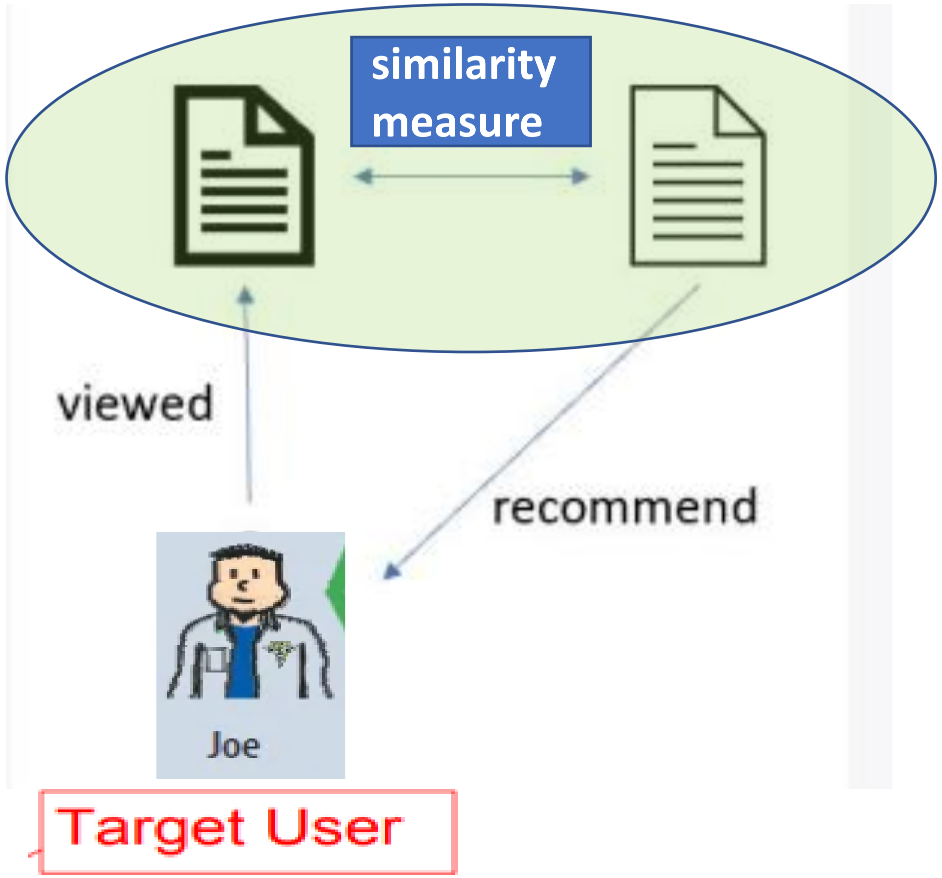
TRADITIONAL APPROACHES

Content-based recommendation. Recommend **items similar** to what a user chose in the past

Collaborative Filtering (CF). Recommends items based on users’ past behavior.

FOCUS: Content-Based Recommendation

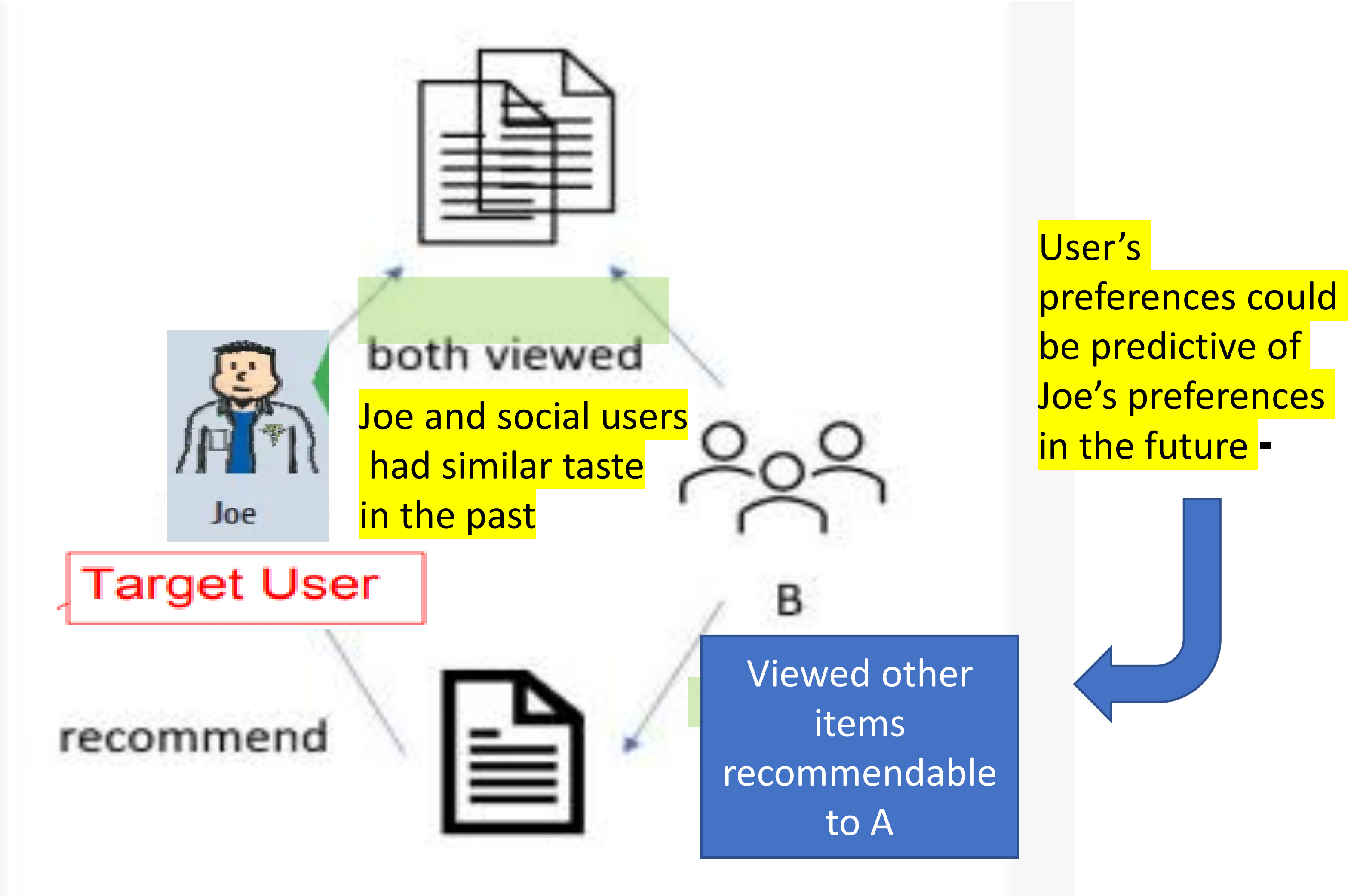
- Recommend items similar to what a user chose in the past
    - Often **uses similarity measure** between items - representations (Salton)



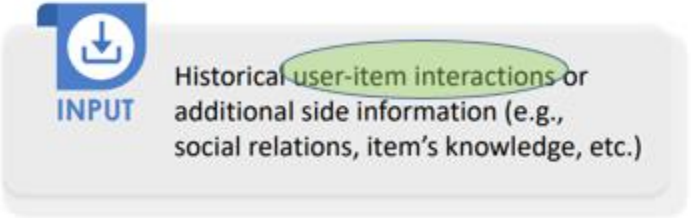
FOCUS: Collaborative filtering

- Past interactions between users and items produce new recommendations.
- Assumptions:

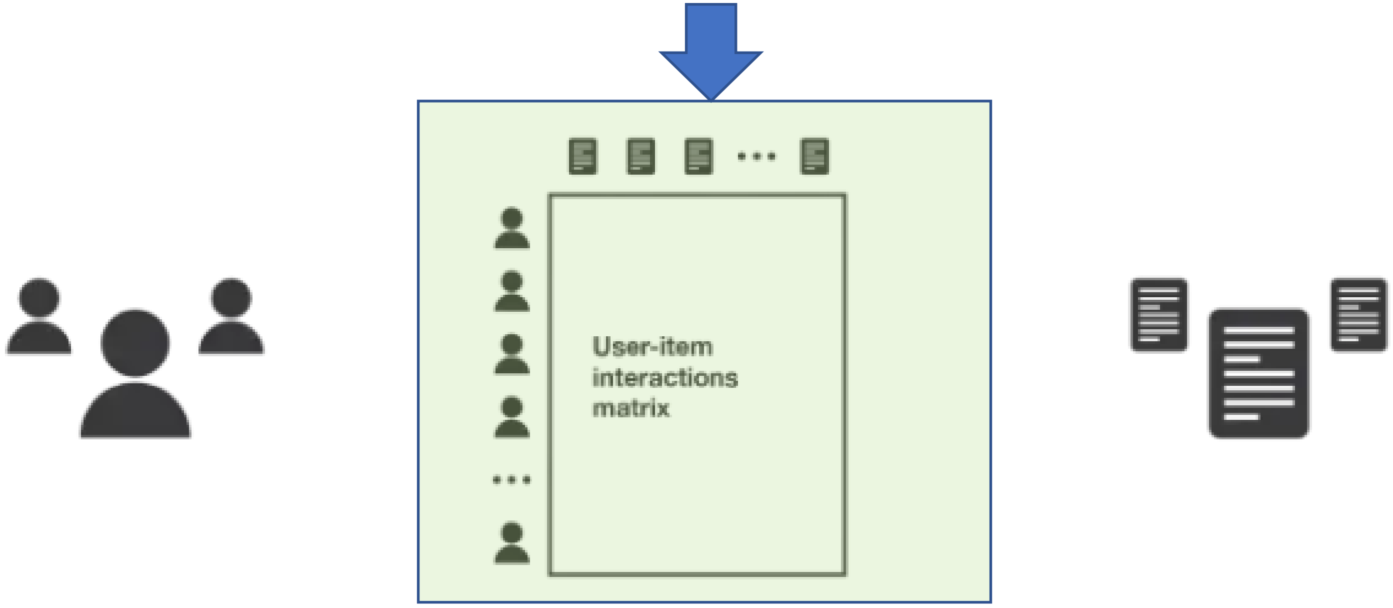
1. Similar users share the same interest and
2. Similar items are liked by a user.



FOCUS: Collaborative filtering



- Typical approach: past user-item interactions (U/I matrix) are stored in the "user-item matrix".



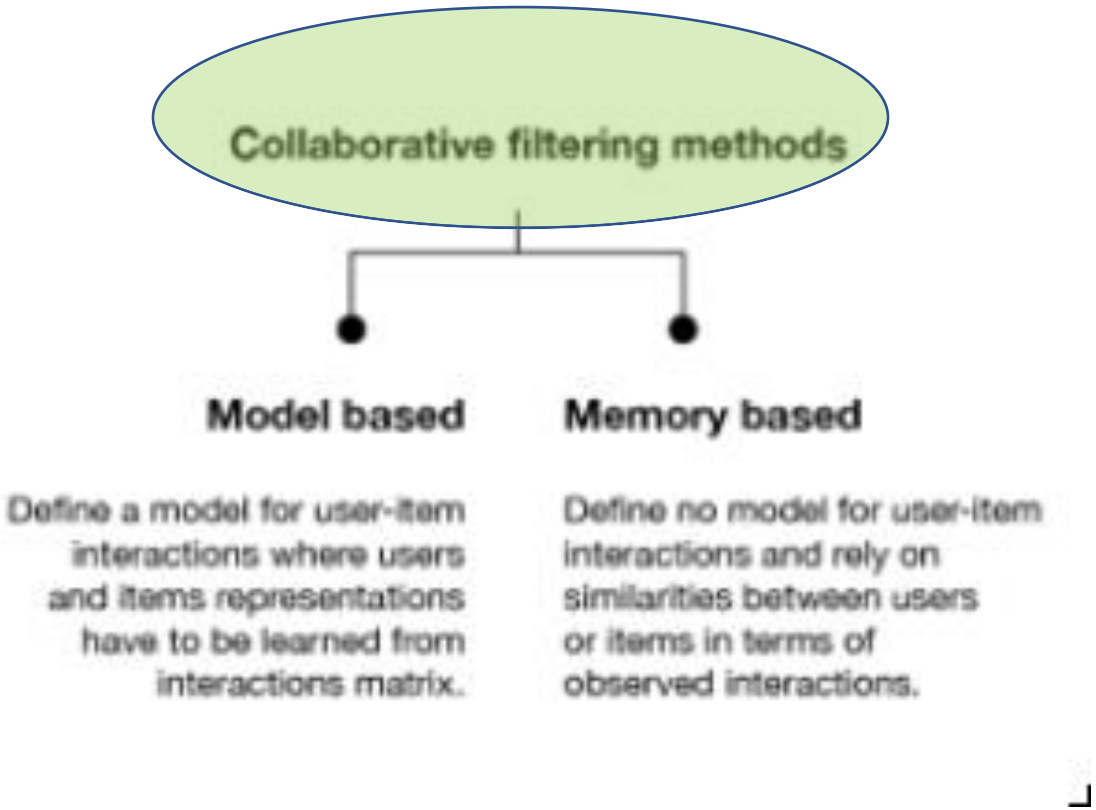
Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products
	...	

user-item interactions matrix provides information to recommends items based on users' past behavior.

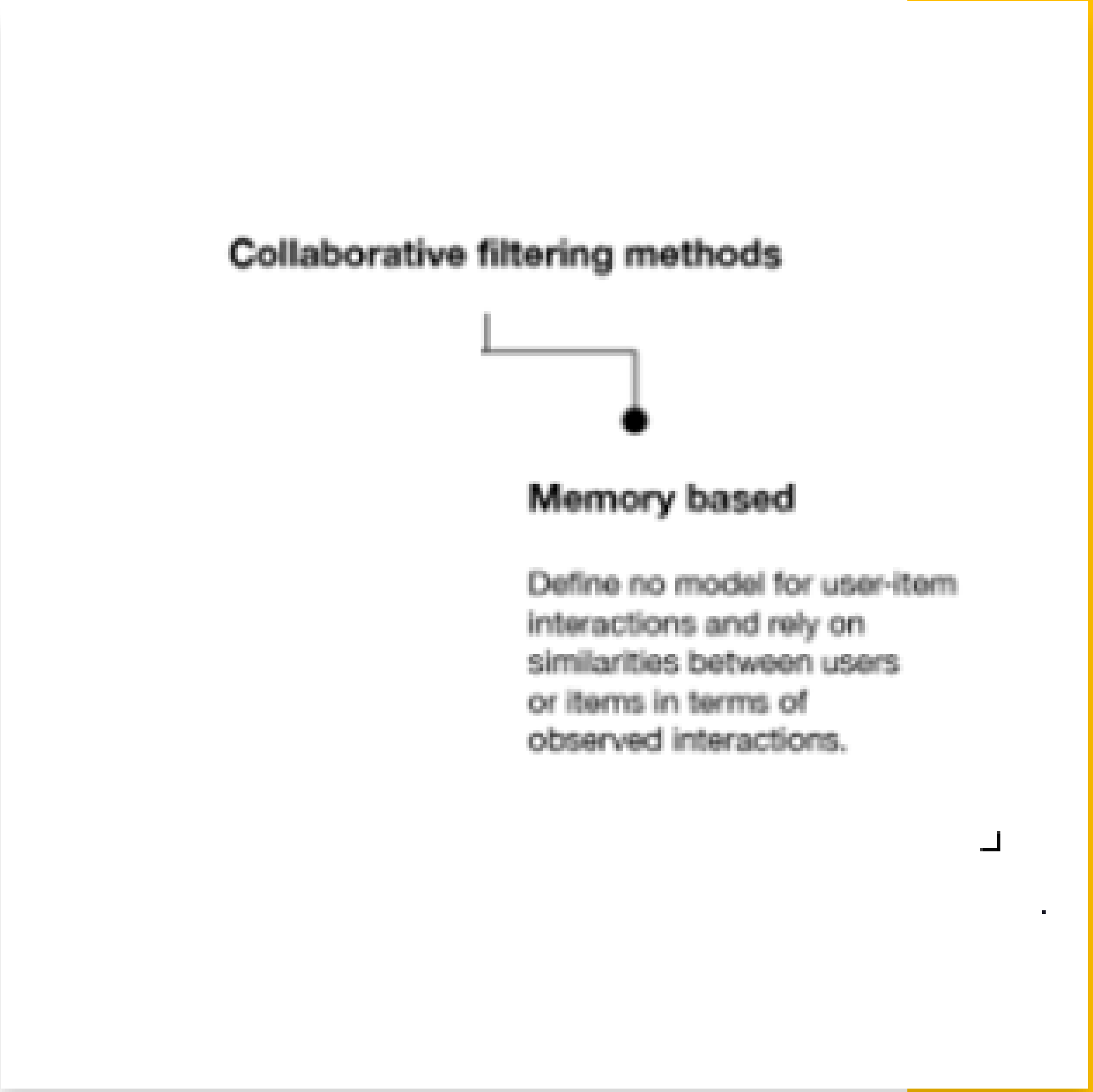
Taxonomy of Collaborative Filtering

- Memory based
  - directly works with values of recorded interactions,
  - no model assumed ,
  - essentially based on nearest neighbours search (e.g., find the closest users from a user of interest and suggest the most popular items among these neighbours).

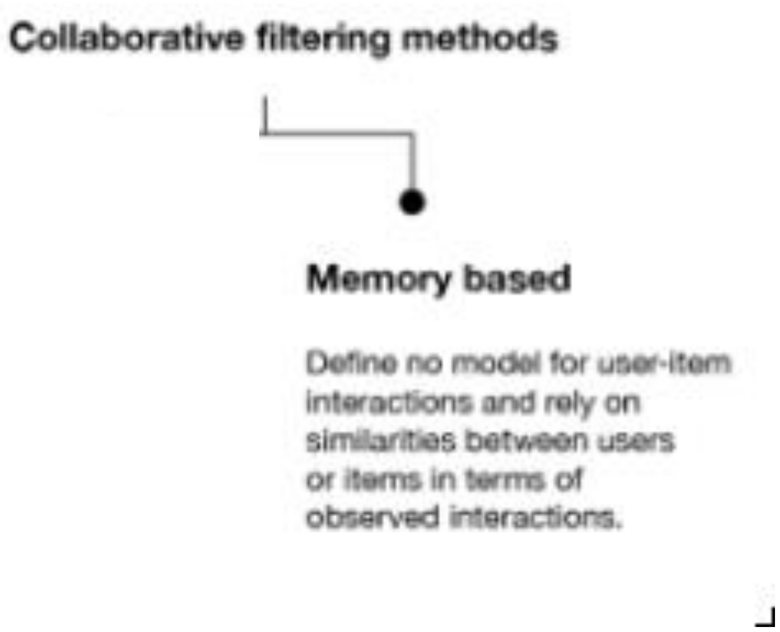
- Model based
  - assume an underlying “generative” model that explains user-item interactions and try to discover it in order to make new predictions.



Memory based CF



Memory Based CF



- There are two categories of Memory Based CF:
  - **User-based**: measure the similarity between target users and other users
  - **Item-based**: measure the similarity between the items that target users rates/ interacts with and other items

User-based CF (“user-centred”)

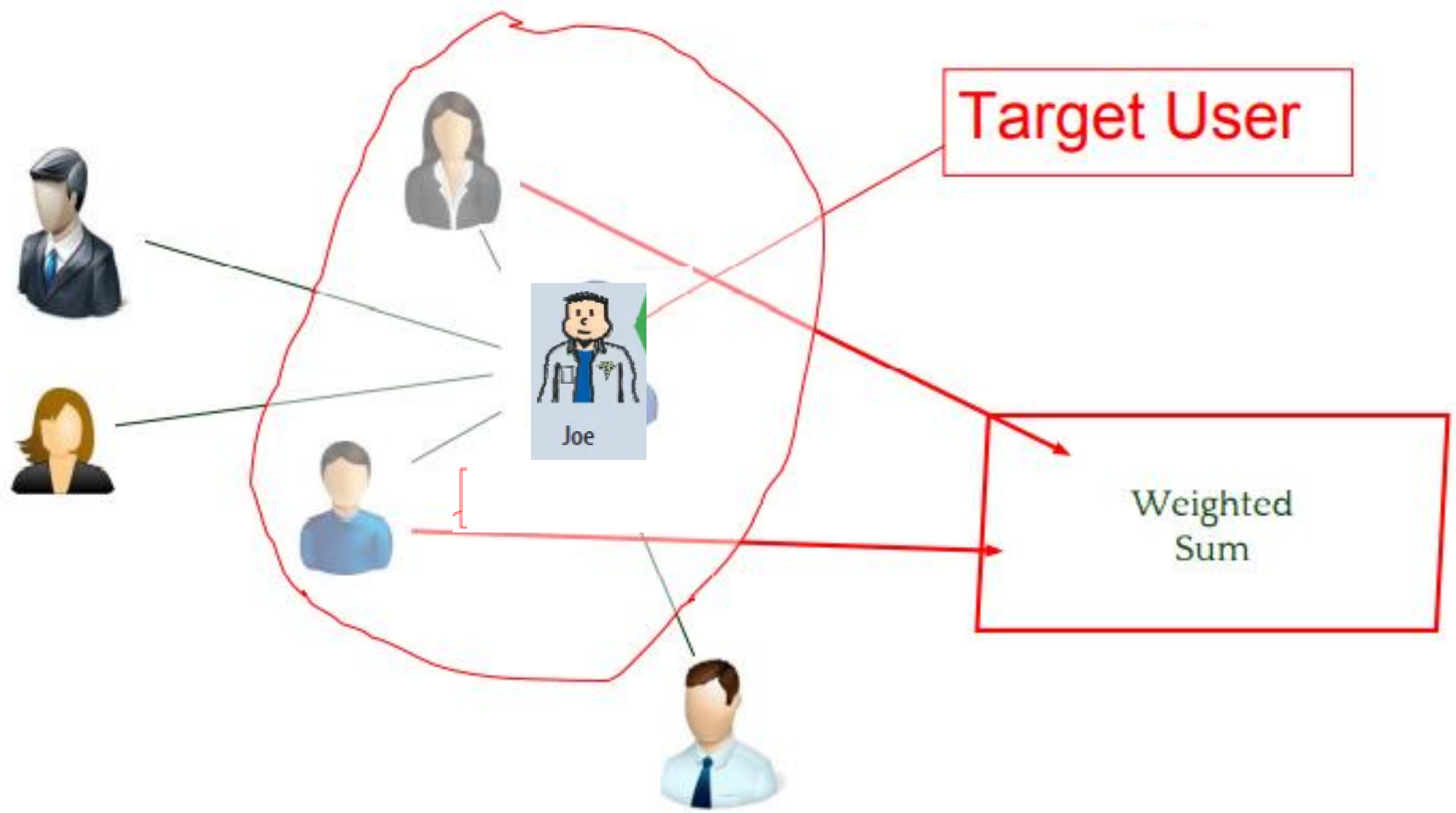
CF assumption: similar users share the same interest and similar items are liked by a user

- tries to identify users with the most similar “interactions profile” (nearest neighbours)
  - in order to suggest items that are the most popular among these neighbours (and that are “new” to our user). It represents users based on their interactions with items and evaluate distances between users.

User-based: intuition

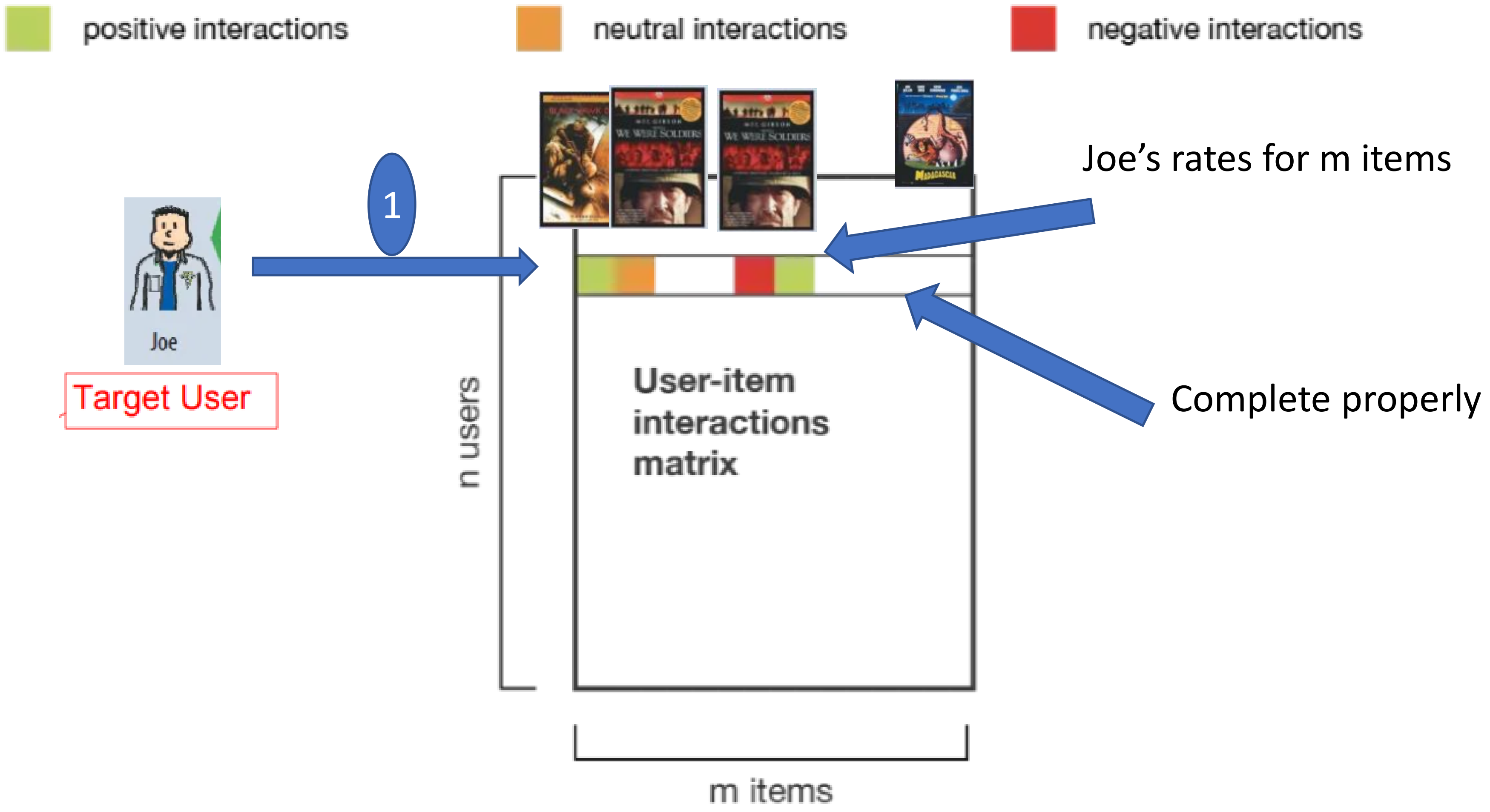
- Consider user u Target User
- Find N other users similar to u, rating-wise (1)
- Predict u' rating based on the other N users. (2)

identify users with the most similar "interactions profile"



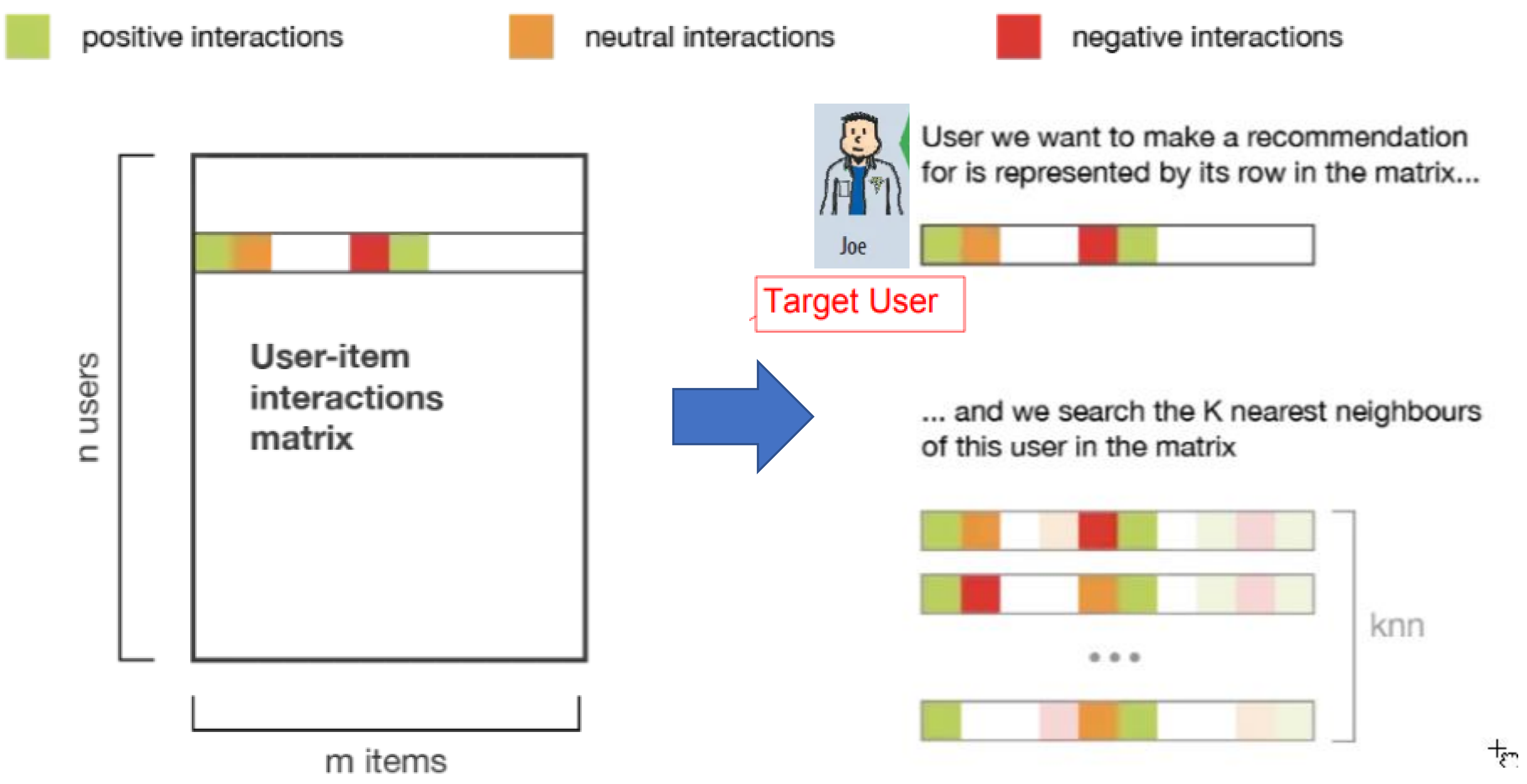
User-based: intuition

- Consider user u Target User
- Find N other users similar to u, rating-wise 1
- Predict u' rating based on the other N users.



User-User CF: Intuition

- Consider user u Target User
- Find N other users similar to u, rating-wise
- Predict u' rating based on the other N users.

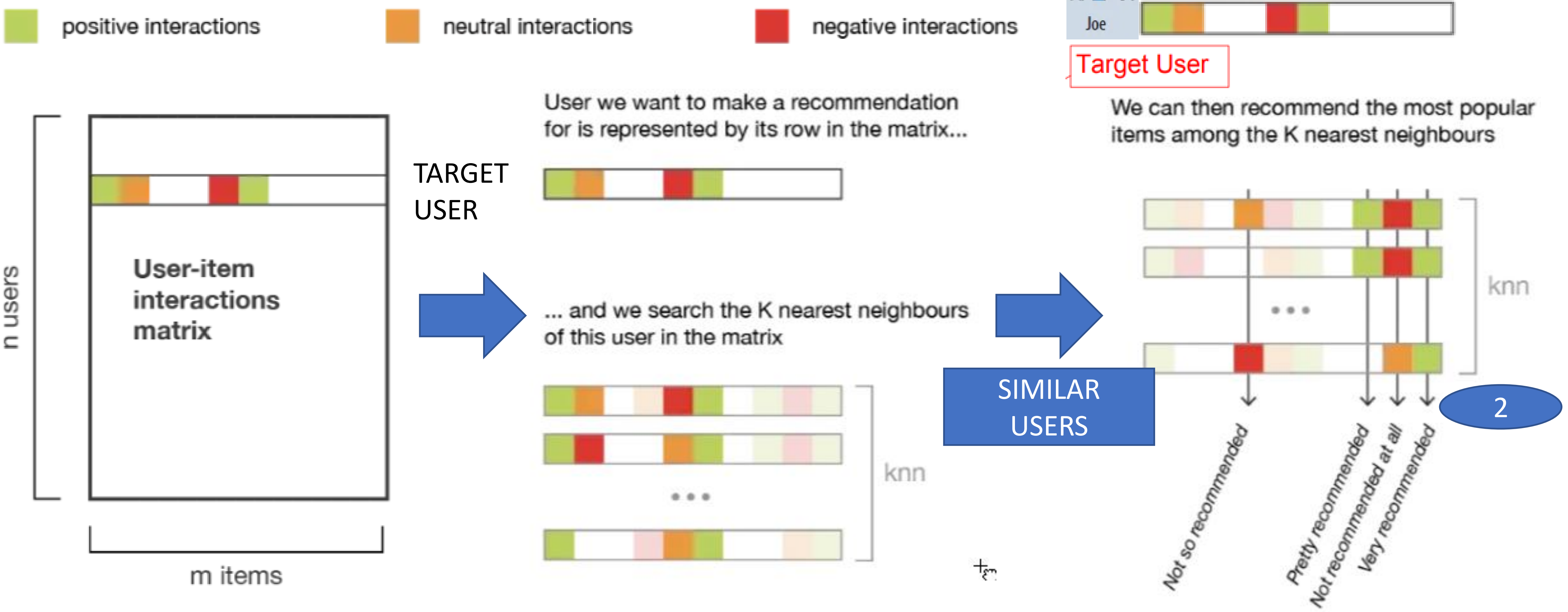


User-User CF: Intuition

- Consider user u Target User
- Find N other users similar to u, rating-wise
- Predict u' rating based on the other N users.

2

Target User



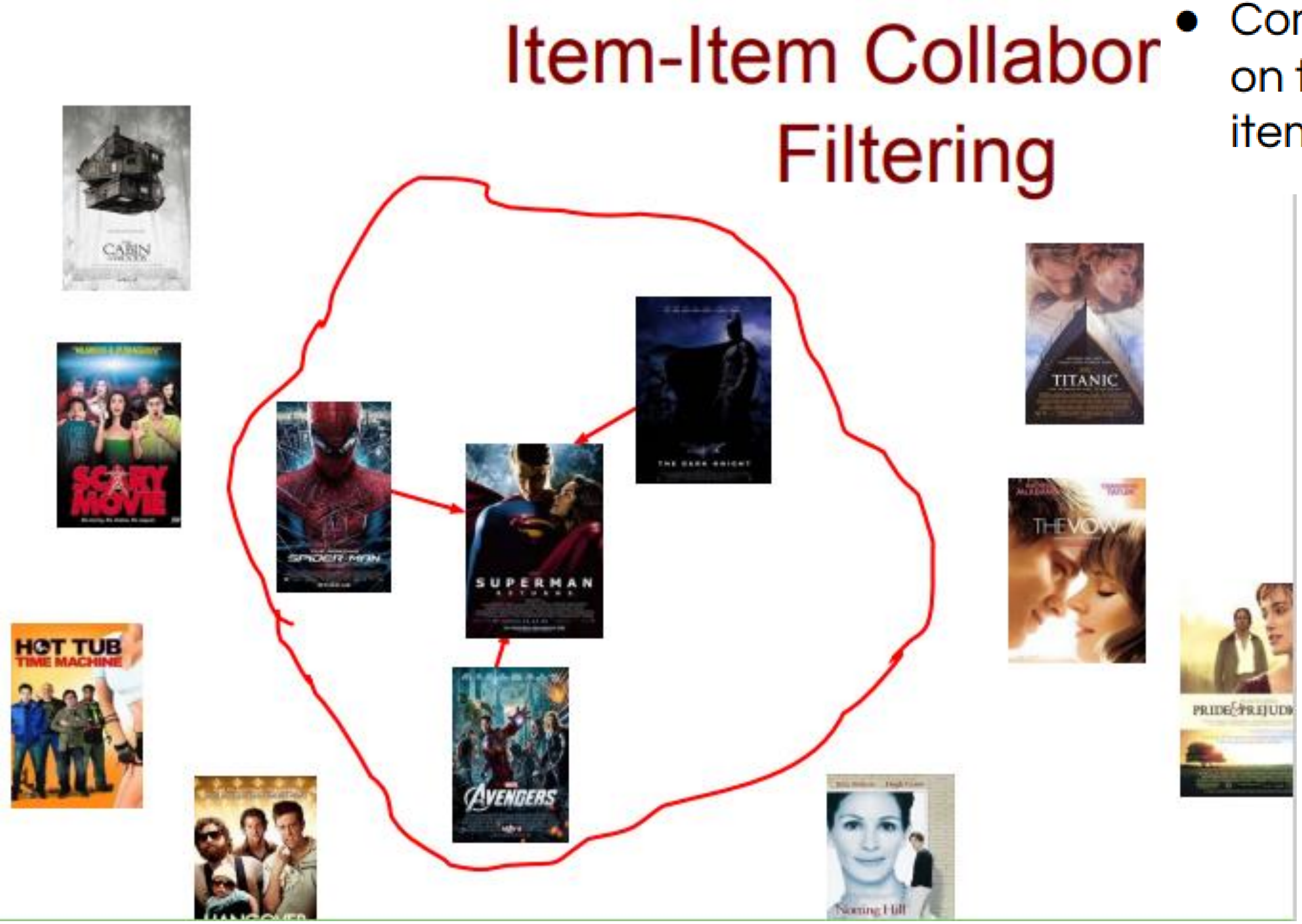
Item- based (“item-centred”)

CF assumption: similar users share the same interest and similar items are liked by a user

- Find items similar to the ones the user already “positively” interacted with.
  - Two items are considered to be similar if most of the users that have interacted with both of them did it in a similar way.

Item- based: intuition

- Look into the items the target user has rated
- Compute how similar they are to the target item
  - Similarity **only using** past **ratings** from other users!
- Select k most similar items.
- Compute Prediction by taking weighted average on the target user's ratings on the most similar items.



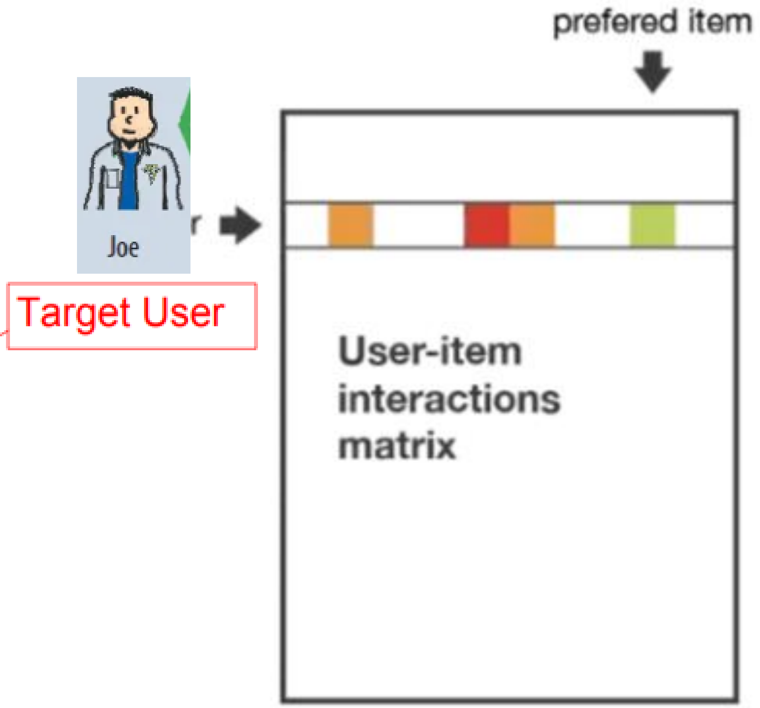
Find items similar to the ones the user already "positively" interacted with.

Item-Item CF: Intuition

- Look into the items the target user has rated
- Compute how similar they are to the target item
  - Similarity **only using** past **ratings** from other users!
- Select k most similar items.
- Compute Prediction by taking weighted average on the target user's ratings on the most similar items.

positive interactions   neutral interactions   negative interactions

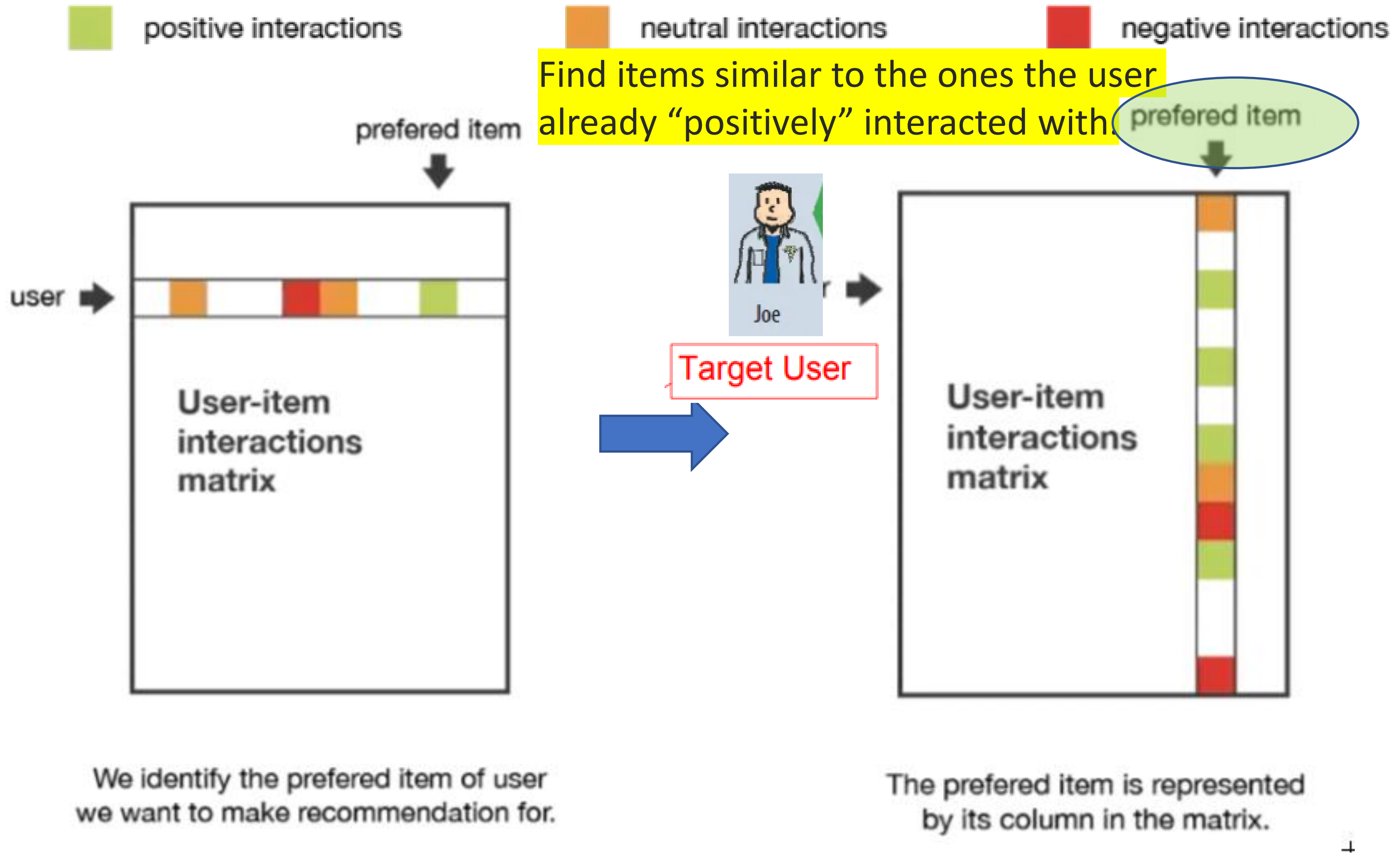
Find items similar to the ones the user already "positively" interacted with.



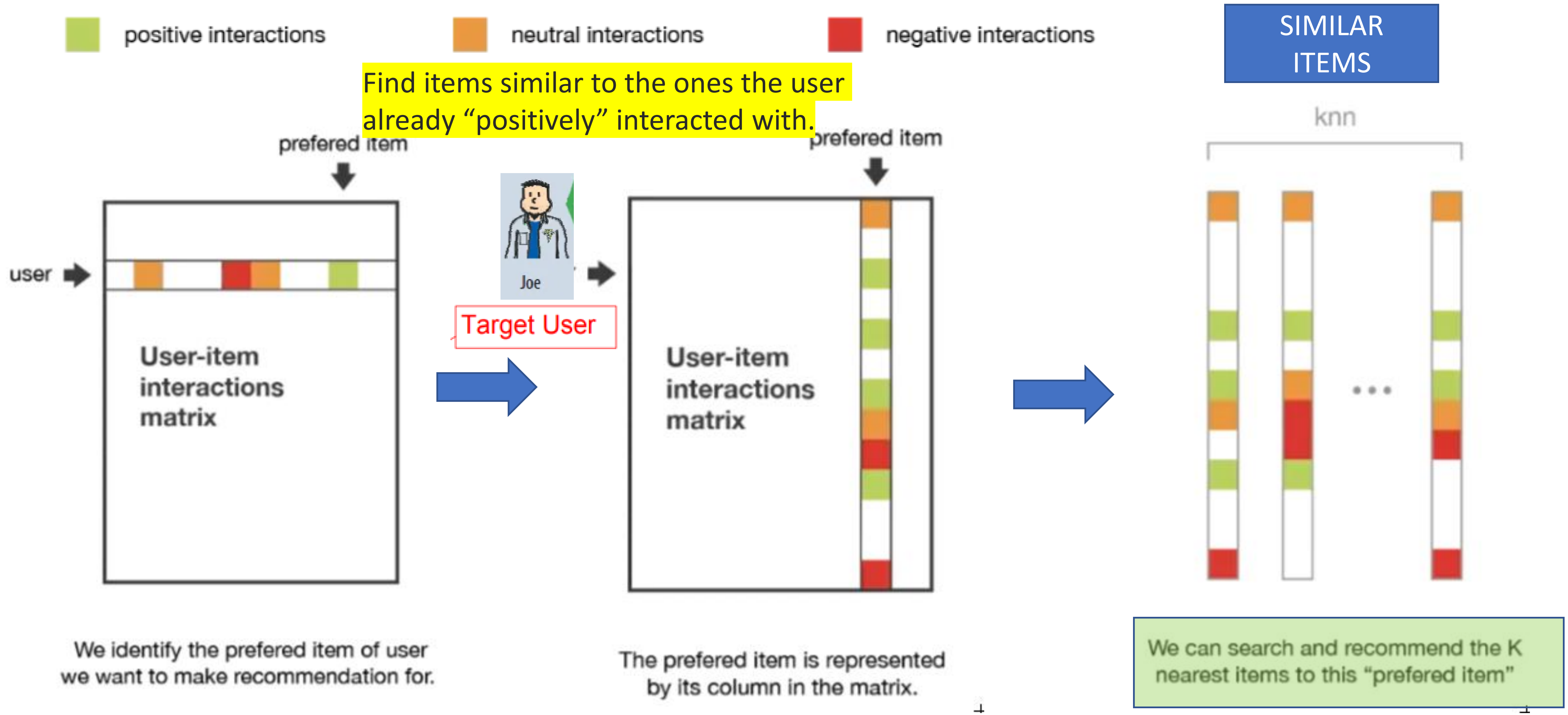
We identify the prefered item of user we want to make recommendation for.

Item-Item CF: Intuition

- Look into the items the target user has rated
- Compute how similar they are to the target item
  - Similarity **only using** past **ratings** from other users!
- Select k most similar items.
- Compute Prediction by taking weighted average on the target user's ratings on the most similar items.



Item-Item CF: Intuition



Recap

- Collaborative recommendation systems
  - “People who agreed in the past are likely to agree in the future”
- Content based
  - ▪ Recommend items similar to what the user liked in the past
- Knowledge based ▪
  - Use domain knowledge to retrieve items that match user needs
- Hybrid ▪
  - Combines above designs. Most common in practice

Can be considered for extended tasks in our lab’s challange !

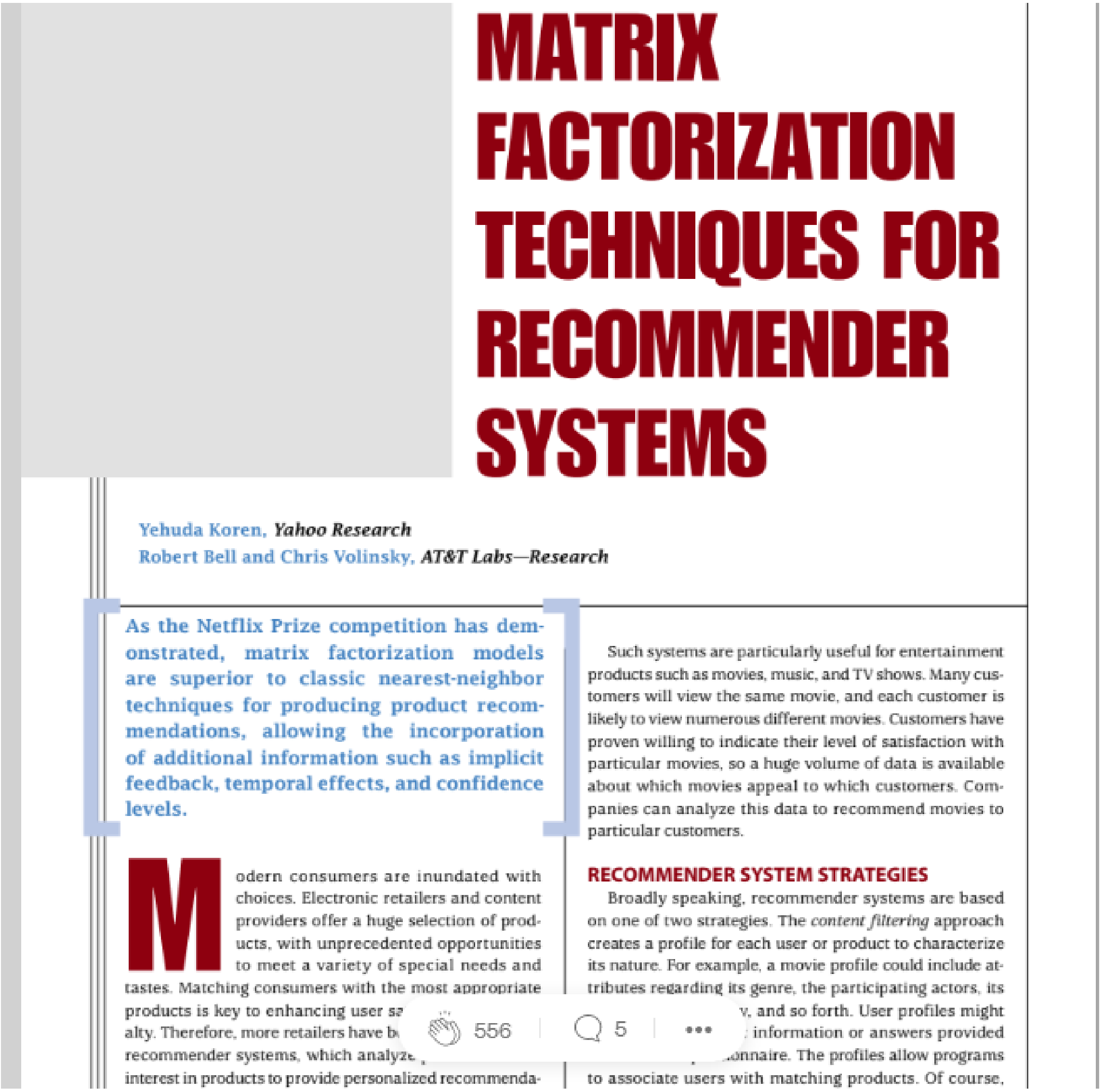
Model Based

Matrix Factorization

- Conventional collaborative filtering model is based on **Matrix Factorization (MF)**.

Matrix factorization for recommandations

- Matrix factorization is an extensively used technique in **collaborative filtering recommendation systems**.
- Objective is to factorize a user-item matrix into two low-ranked matrices, the user-factor matrix and the item-factor matrix



MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS

Yehuda Koren, *Yahoo Research*
Robert Bell and Chris Volinsky, *AT&T Labs—Research*

As the Netflix Prize competition has demonstrated, matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels.

Modern consumers are inundated with choices. Electronic retailers and content providers offer a huge selection of products, with unprecedented opportunities to meet a variety of special needs and tastes. Matching consumers with the most appropriate products is key to enhancing user sa… alty. Therefore, more retailers have b… recommender systems, which analyze … interest in products to provide personalized recommenda-

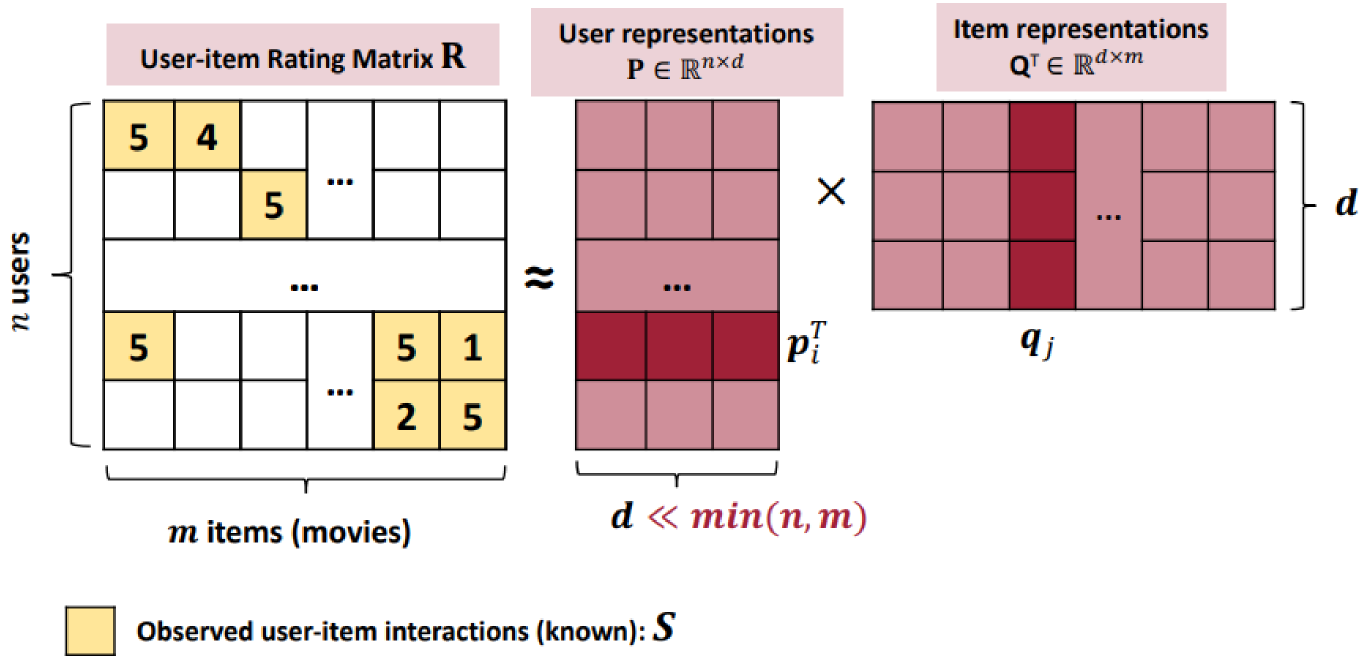
Such systems are particularly useful for entertainment products such as movies, music, and TV shows. Many customers will view the same movie, and each customer is likely to view numerous different movies. Customers have proven willing to indicate their level of satisfaction with particular movies, so a huge volume of data is available about which movies appeal to which customers. Companies can analyze this data to recommend movies to particular customers.

RECOMMENDER SYSTEM STRATEGIES

Broadly speaking, recommender systems are based on one of two strategies. The *content filtering* approach creates a profile for each user or product to characterize its nature. For example, a movie profile could include attributes regarding its genre, the participating actors, its …, and so forth. User profiles might … information or answers provided …onnaire. The profiles allow programs to associate users with matching products. Of course,

Matrix factorization

- matrix factorization can be thought of as finding 2 matrices whose product is the original matrix.

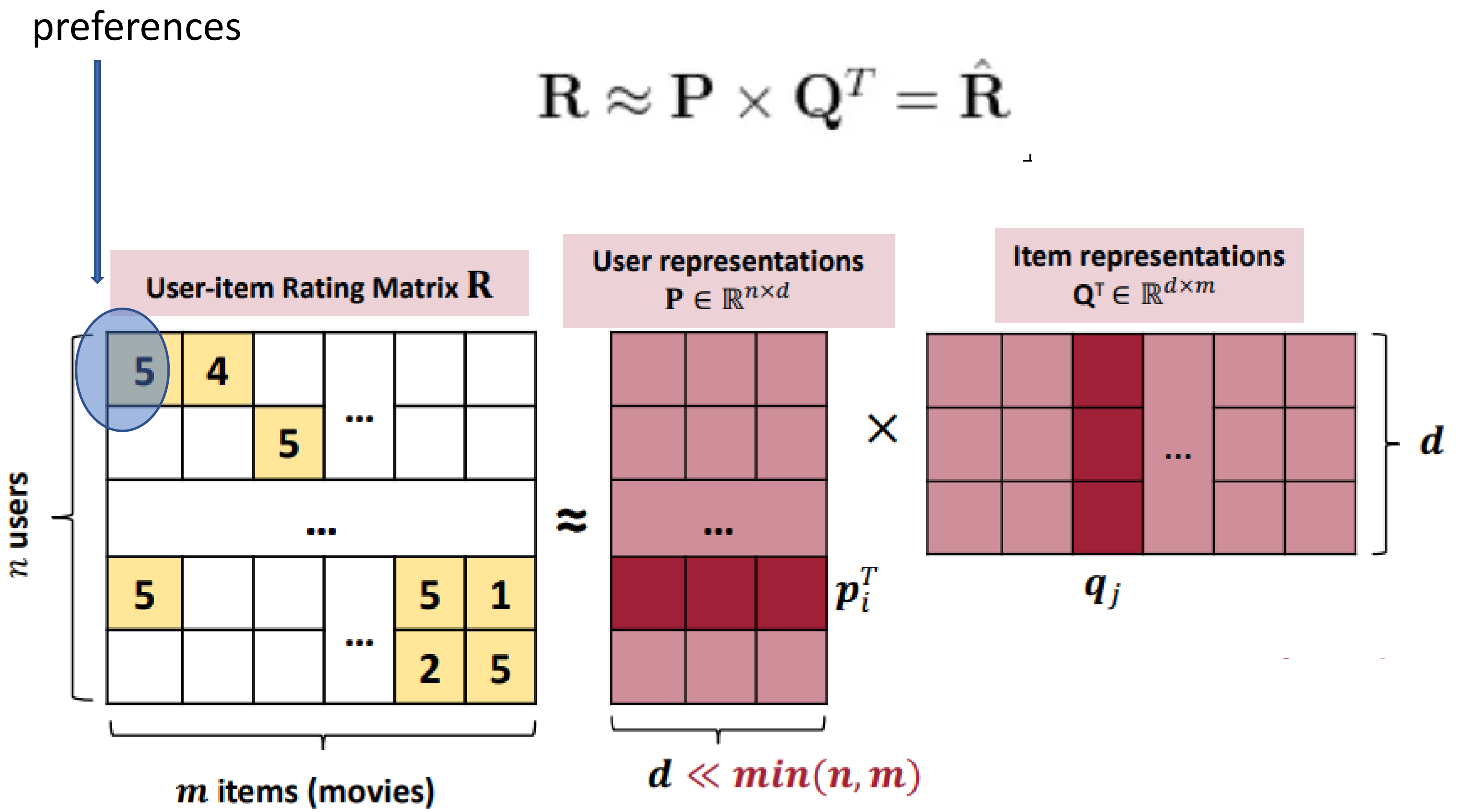


$$R \approx P \times Q^T = \hat{R}$$

Main idea

- Assume there exists a latent space of features in which we can represent both users and items and
- such that the interaction between a user and an item can be obtained by computing the dot product of corresponding dense vectors in that space.

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$



Generate latent features when multiplying two different kinds of entities.

More formally

- user vector represents their preferences, item vector represents its features, dot product measures similarity
- Find d-length vectors for each user and item such that

Task: rating prediction in Netflix

Given $n \times m$ matrix $\mathbf{R}$, the goal is to learn:

Users/Items representations: $\mathbf{P} \in \mathbb{R}^{n\times d}$, $\mathbf{Q} \in \mathbb{R}^{m\times d}$

Objective with rating reconstruction error:

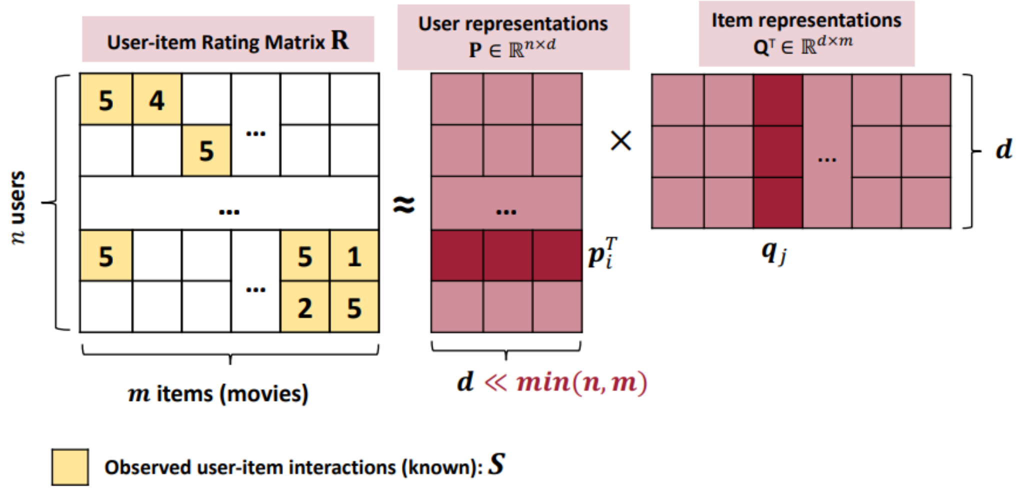
$$\min_{\mathbf{P},\mathbf{Q}} \sum_{i,j\in S} (r_{ij} - \hat{r}_{ij})^2 = \sum_{i,j\in S} (r_{ij} - \boldsymbol{p}_i^T \boldsymbol{q}_j)^2$$

observed rating score

predicted rating score

9

We want the error as lower as possible by properly fitting these parametes

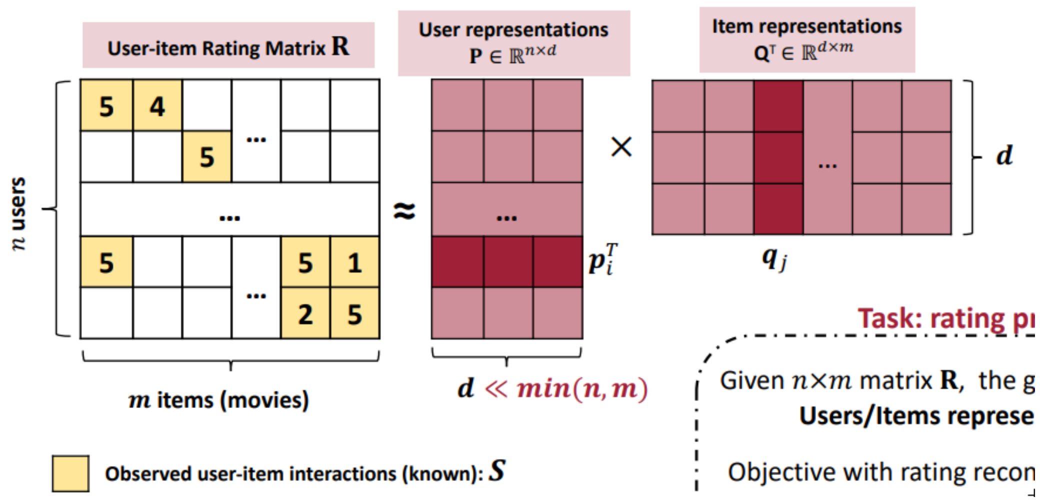


$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

Matrix factorization learning: Notation

- One obvious method to find matrix q and p is the gradient descent method.

- u – user, i – item
- r_{ui} – rating
- $\hat{r}_{ui}$ – predicted rating
- b, b_u, b_i – bias
- q_i, p_u – latent factor vectors (length k)



Assumptions and notation

- Prediction $\hat{r}_{ij}$ for $<$ *user*, *item* $>$ pair i, j :

(for a while assume centered data without bias)

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

- vector multiplication
- user-item interaction via latent factors

illustration (3 factors):

- user (p_u): $(0.5, 0.8, -0.3)$
- item (q_i): $(0.4, -0.1, -0.8)$

- u – user, i – item
- r_{ui} – rating
- $\hat{r}_{ui}$ – predicted rating
- b, b_u, b_i – bias
- q_i, p_u – latent factor vectors (length k)

Loss formulation

- u – user, i – item
- r_{ui} – rating
- $\hat{r}_{ui}$ – predicted rating
- b, b_u, b_i – bias
- q_i, p_u – latent factor vectors (length k)

- Prediction $\hat{r}_{ij}$ for < *user, item* > pair i, j :

(for a while assume centered data without bias)

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

- we want to minimize "squared errors" (related to RMSE, more details leater)

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

We want the error as lower as possible by properly fitting these parametes

Problem formulation and regularization

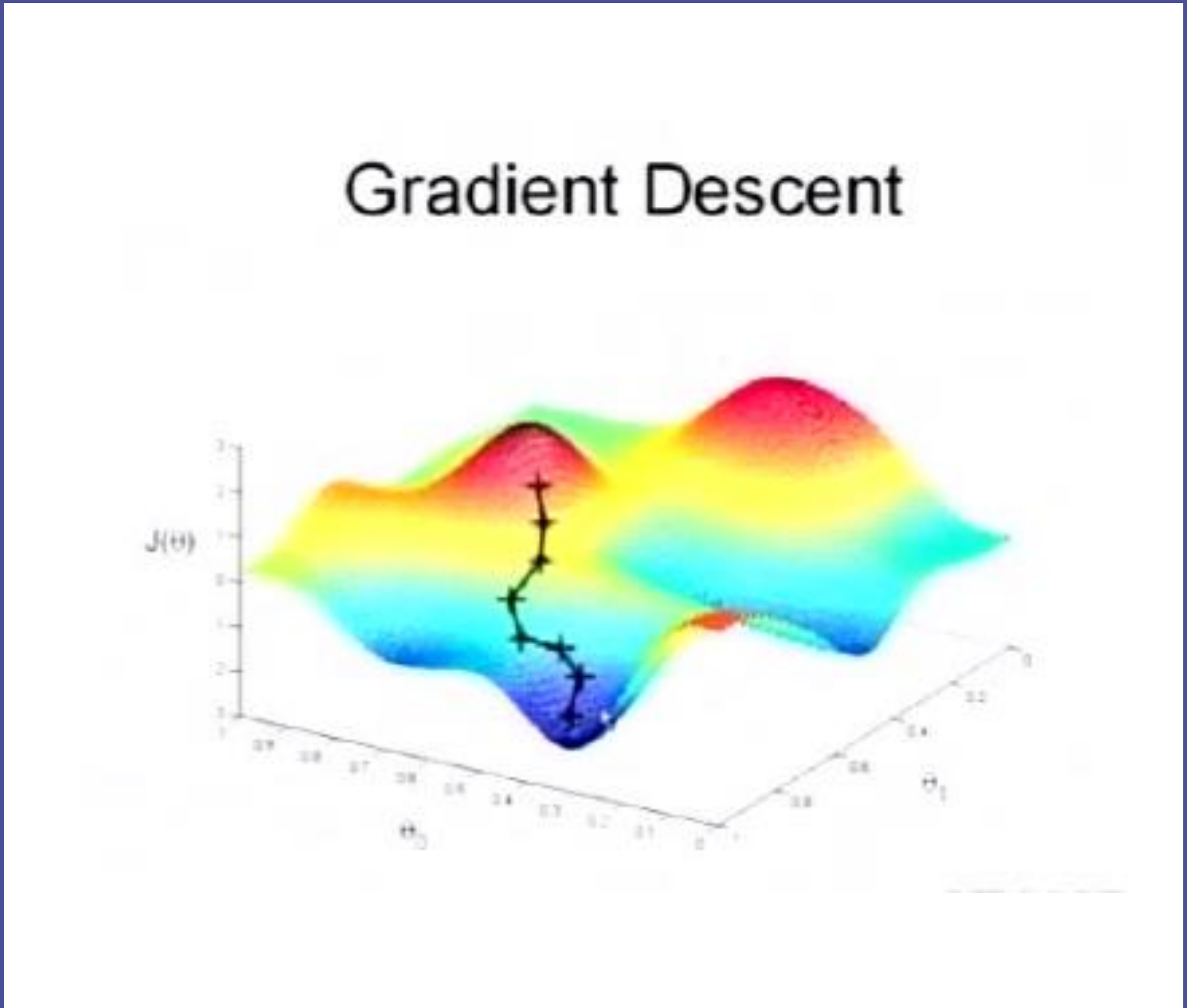
- regularization to avoid overfitting (standard machine learning approach)

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

- we want to minimize "squared errors" (related to RMSE, more details leater)

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

How to find the minimum?



How to find the minimum?

- we want to minimize “squared errors” (related to RMSE, more details leater)

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

Matrix factorization learning

- we want to minimize "squared errors" (related to RMSE, more details leater)

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

- Loss for prediction where true rating is r_{ij}:

$$L(r_{ij}, \hat{r}_{ij}) = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2$$

Let me use matrix notation here

Loss differentiation ....

- we want to minimize "squared errors" (related to RMSE, more details leater)

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

- Loss for prediction where true rating is r_{ij}:

$$L(r_{ij}, \hat{r}_{ij}) = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2$$

Let me use matrix notation here

- Gradient of loss function for sample $< i, j >$:

$$\frac{\partial L(r_{ij}, \hat{r}_{ij})}{\partial U_{if}} = \frac{\partial (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2}{\partial U_{if}} = -2(r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf}) V_{jf}$$

$$\frac{\partial L(r_{ij}, \hat{r}_{ij})}{\partial V_{jf}} = \frac{\partial (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2}{\partial V_{jf}} = -2(r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf}) U_{if}$$

  - for $f = 1$ to F

Loss differentiation ....

- Gradient of loss function for sample $< i, j >$:

$$\frac{\partial L(r_{ij}, \hat{r}_{ij})}{\partial U_{if}} = \frac{\partial (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2}{\partial U_{if}} = -2(r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf}) V_{jf}$$

$$\frac{\partial L(r_{ij}, \hat{r}_{ij})}{\partial V_{jf}} = \frac{\partial (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2}{\partial V_{jf}} = -2(r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf}) U_{if}$$

  - for $f = 1$ to F

- Let's simplify the notation:

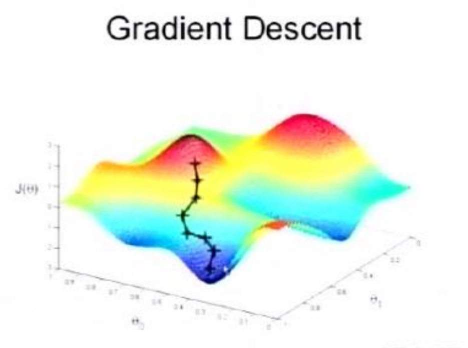
$$\text{let } e = r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf} \qquad \text{(the prediction error)}$$

$$\frac{\partial L(r_{ij}, \hat{r}_{ij})}{\partial U_{if}} = \frac{\partial e^2}{\partial U_{if}} = -2eV_{jf}$$

$$\frac{\partial L(r_{ij}, \hat{r}_{ij})}{\partial V_{jf}} = \frac{\partial e^2}{\partial V_{jf}} = -2eU_{if}$$

  - for $f = 1$ to F

Gradient descent



1. Decide on F = dimension of factors
2. Initialize factor matrices with small random values
3. Choose one sample from training set
4. Calculate loss function for that single sample
5. Calculate gradient from loss function
6. Update $2 \cdot F$ model parameters a single step using gradient and learning rate
7. Repeat from 3) until stopping criterion is satisfied

- Set learning rate = η
- Then the factor matrix updates for sample $< i, j >$ are:

$$U_{if} = U_{if} + 2\eta e V_{jf}$$
$$V_{jf} = V_{jf} + 2\eta e U_{if}$$

  - for $f = 1$ to F

Gradient descent with the regularized step

- Must use some form of regularization (usually L_2):

$$L(r_{ij},\hat{r}_{ij}) = (r_{ij} - \sum_{f=1}^{F} U_{if} \cdot V_{jf})^2 + \lambda \sum_{f=1}^{F} U_{if}{}^2 + \lambda \sum_{f=1}^{F} V_{jf}{}^2$$

- Update rules become:

$$\boxed{\begin{aligned} U_{if} &= U_{if} + 2\eta(eV_{jf} - \lambda U_{if}) \\ V_{jf} &= V_{jf} + 2\eta(eU_{if} - \lambda V_{jf}) \end{aligned}}$$

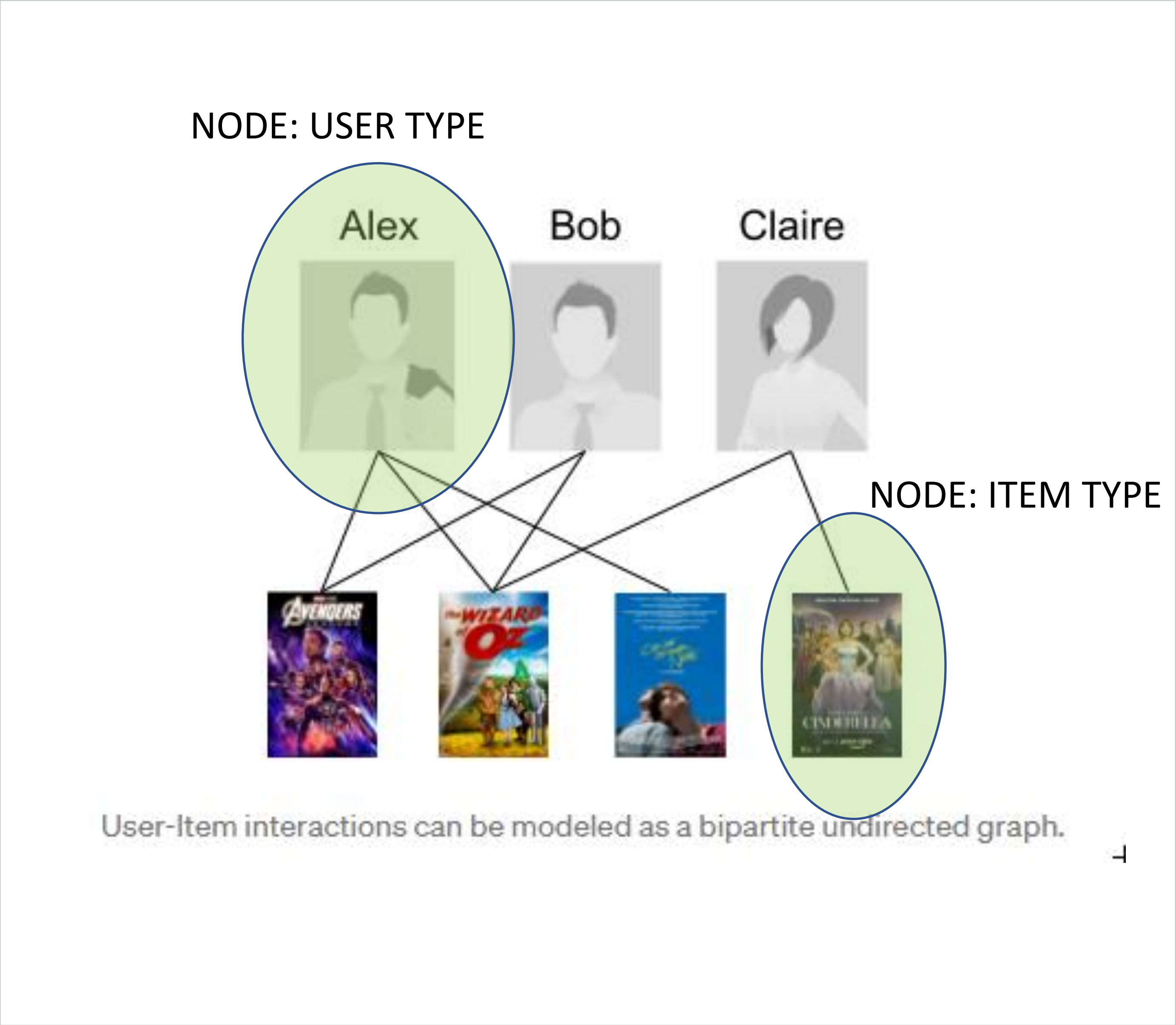
  - for $f = 1$ to F

Deep recommender systems

A deep model based RS!

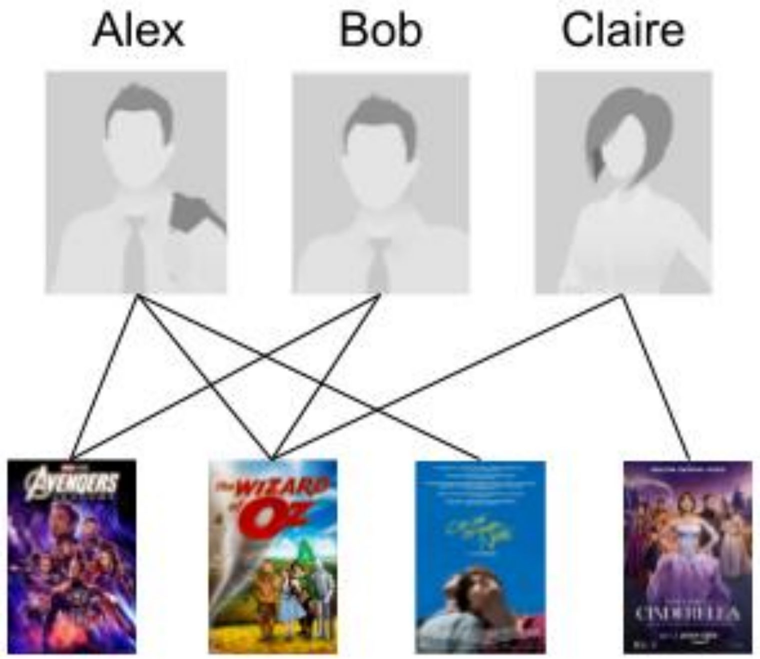
User item relationships as bipartite graphs

- Recommender system can be naturally modelled as a bipartite graph:
- Two types of nodes: **users** and **items**.
    - Edges connect users and items Indicates user-item interaction (e.g., click, purchase, review etc.)
    - Often associated with timestamp (timing of the interaction)



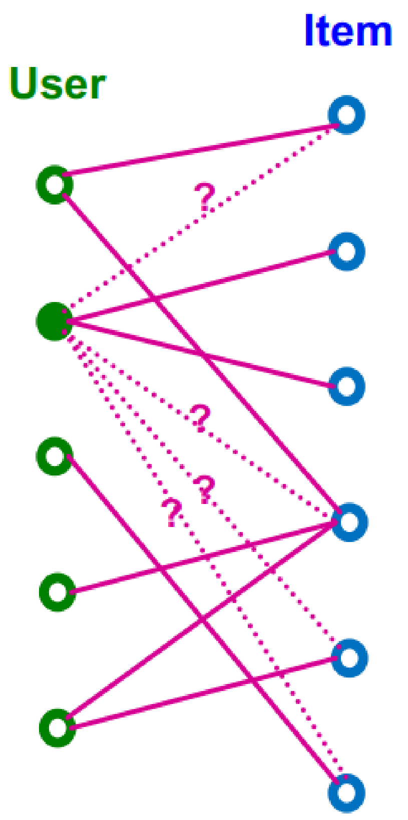
Notation

- **Notation**:
  - $\boldsymbol{U}$: A set of all users
  - $\boldsymbol{V}$: A set of all items
  - $\boldsymbol{E}$: A set of observed user-item interactions
    - $\boldsymbol{E} = \{(u, v) \mid u \in \boldsymbol{U}, v \in \boldsymbol{V}, u \text{ interacted with } v\}$



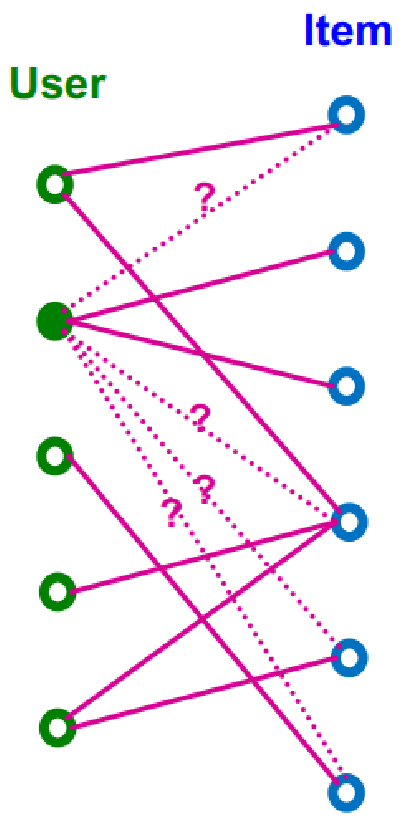
User-Item interactions can be modeled as a bipartite undirected graph.

Problem formulation



- **Given**
  - Past user-item interactions
- **Task**
  - Predict new items each user will interact in the future.
  - Can be cast as **link prediction** problem.
    - Predict new user-item interaction edges given the past edges.

Problem formulation

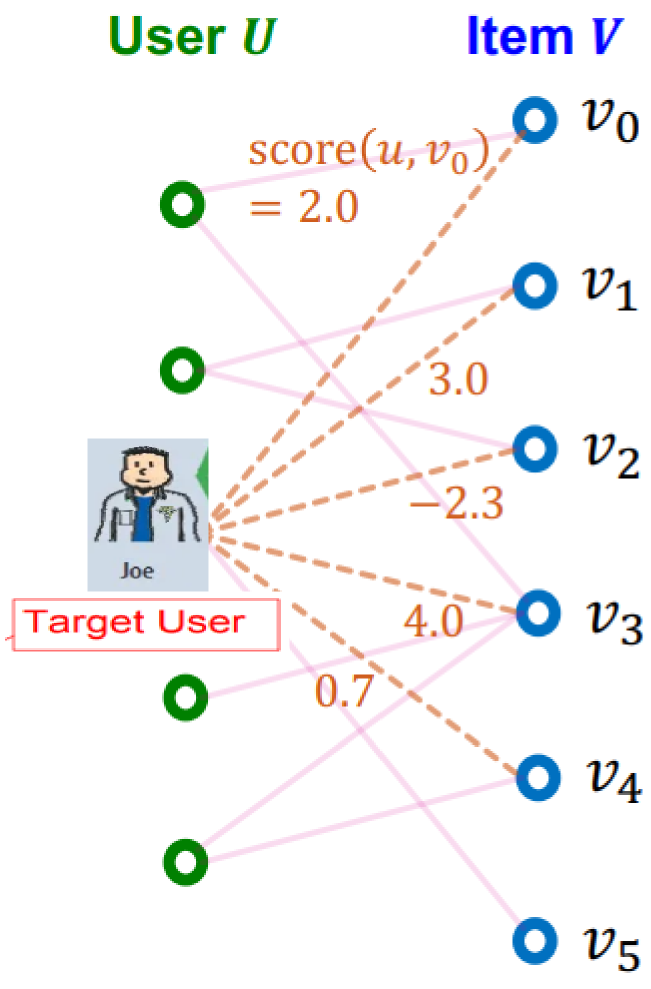


- For each user, we recommend K items.
  - **In order for recommendation to be effective, K needs to be much smaller than the total number of items (up to billions)**
  - K is typically in the order of 10—100.
- The goal is to include as many **positive items** as possible in the top-K recommended items.
  - **Positive items = Items that the user will interact with in the future.**

TARGET

For each user, we recommend K items.

- To get the top-K items, we need a score function for user-item interaction:
  - For $u \in \boldsymbol{U}, v \in \boldsymbol{V}$, we need to get a real-valued scalar $\text{score}(u, v)$.
  - **K items with the largest scores for a given user u** (excluding already-interacted items) are then recommended.

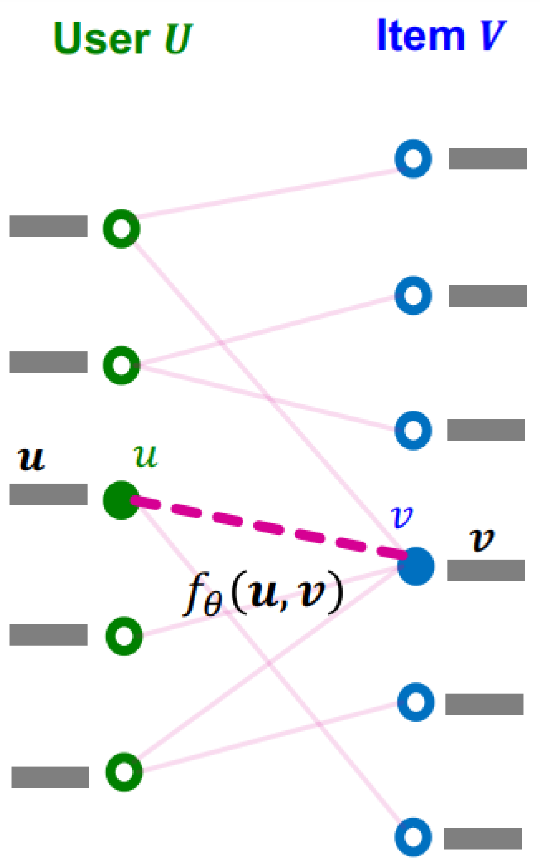


For $K = 2$, recommended items for user u would be $\{v_1, v_3\}$.

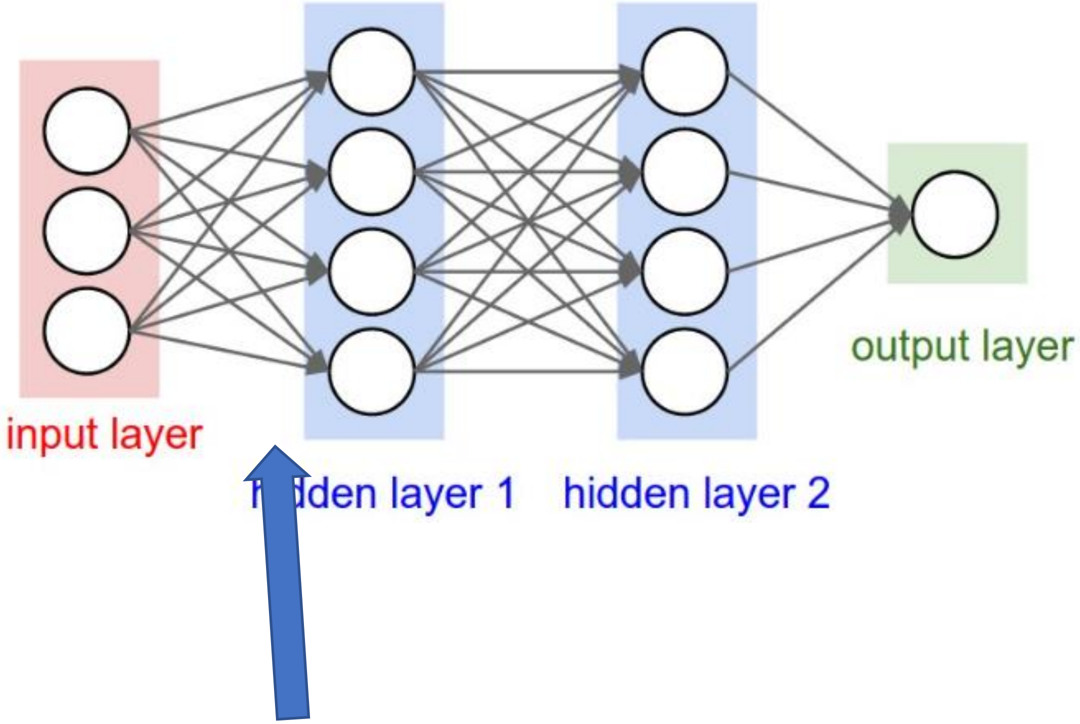
EMBEDDING based k top scores

- We consider **embedding-based models** for scoring user-item interactions.
  - For each user $u \in \boldsymbol{U}$, let $\boldsymbol{u} \in \mathbb{R}^D$ be its D-dimensional embedding.
  - For each item $v \in \boldsymbol{V}$, let $\boldsymbol{v} \in \mathbb{R}^D$ be its D-dimensional embedding.
  - Let $f_\theta(\cdot,\cdot): \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a parametrized function.
  - Then, $\boxed{\text{score}(u, v) \equiv f_\theta(\boldsymbol{u}, \boldsymbol{v})}$

Can be a NN

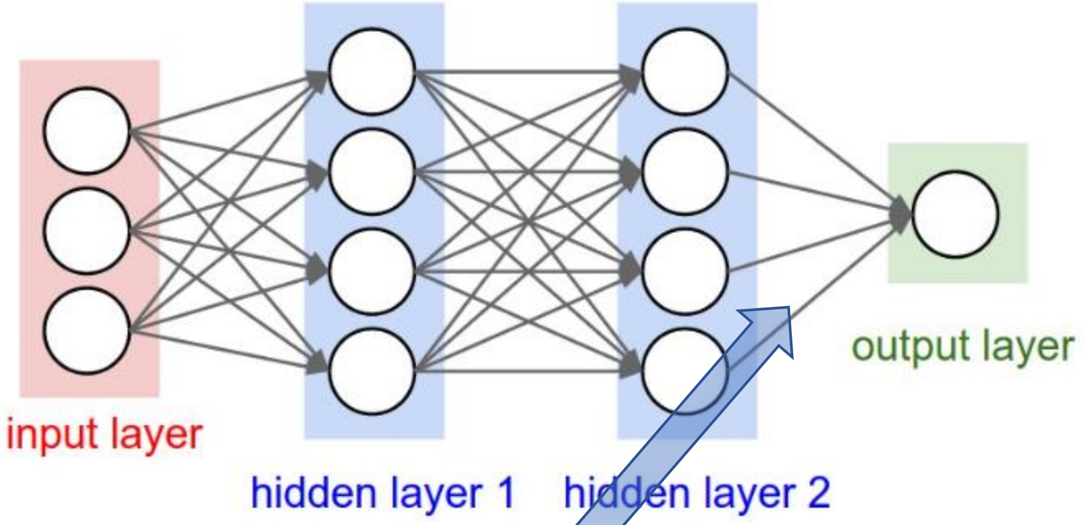


Remarks: Embedding ?



Embedding from input

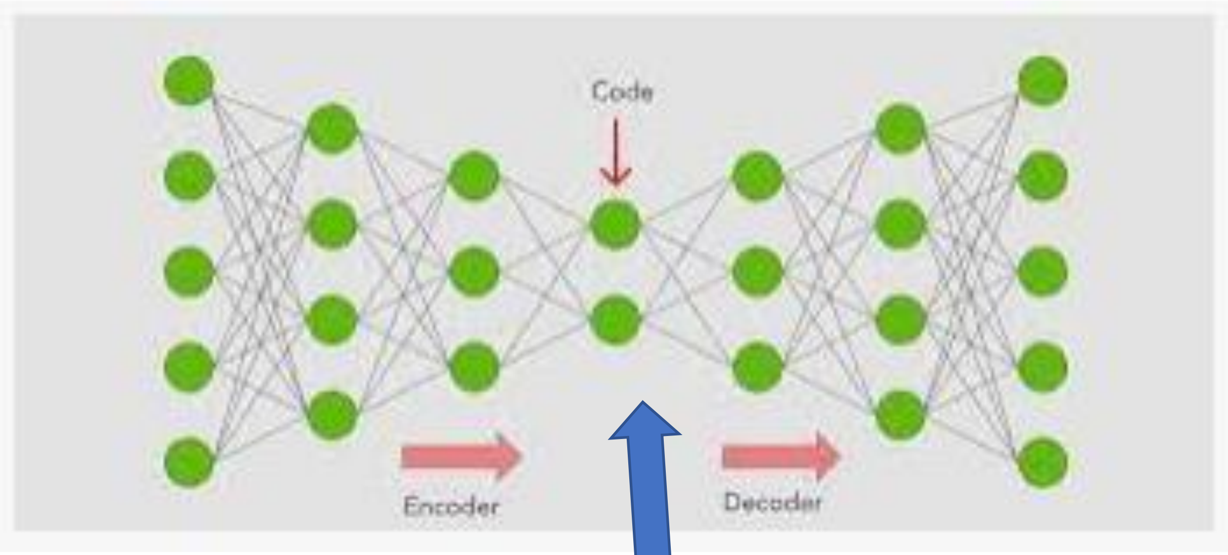
Remarks: Embedding ?



Another latent representation from hidden layer n2

$h2 = \sigma(h1)$

Remarks: Embedding with autoencoder

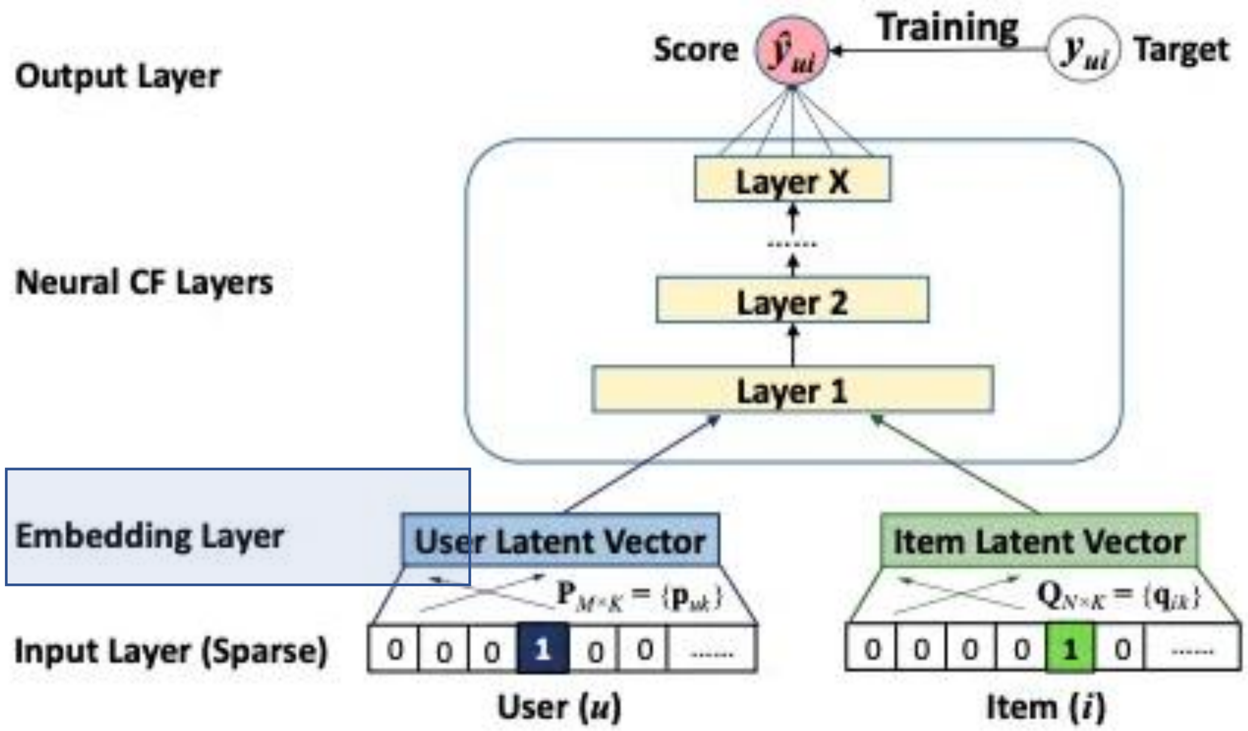


Embedding from input

A naive deep mechanism

- We consider **embedding-based models** for scoring user-item interactions.

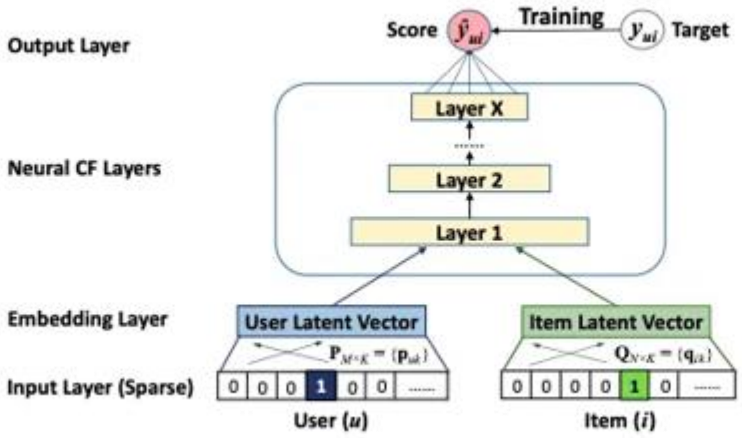
dense or latent vectors for the sparse inputs



Basically both items and users are one-hot encoded.

Fig.2 from "Neural Collaborative Filtering" by X He, L Liao, H Zhang, L Nie, X Hu, TS Chua — Proceedings of the 26th international conference on world wide web, 2017

Remark: A deep mechanism



- The model itself does ***not explicitly*** capture graph structure
  - The graph structure is ***only implicitly*** captured in the training objective.

Some related issues with RS

Problem: Serendipity

- You watch a cat video ->
  - get recommended more cat videos ->
    - watch more cat videos ->
      - recommended more cat videos...

- Interesting at first but becomes repetitive: never explores new interests
- Potential solutions: -
  - Look at how similar users shifted interests in the past –
- Domain knowledge that “if user needs X, they are more likely to need Y in the future”

Problem: Cold Start

- User doesn't have an informative history of interactions on the platform –
- New user or the interaction is sparse by nature (buying cars) –
- Potential solution: - Query the user for more information - E.g. import contacts when signing up for social media, ask what type of car (size, # seats, model, etc.) the user needs

Problem: Feature Extraction

- So far, we had to manually pick out the features that we think are relevant –
- Manual labelling becomes infeasible when –
  - There are simply too many items –
  - Items cannot be described by a few discrete labels. for example, images and videos –
- Potential solution: - Neural methods that can learn useful features from data

More problems

- Scalability - how to create efficient recommendation systems that work with millions or billions of users and items, when pairwise metrics are infeasible
- Large Datasets - how to process large amounts of data, which is possibly unstructured Sparse Ratings - how to create confident recommendations to users about items even if little information was provided about those items Lack of user profiles –

References

- Koren, Yehuda, Robert Bell, and Chris Volinsky. *Matrix factorization techniques for recommender systems.* Computer 42.8 (2009): 30-37.
- Koren, Yehuda, and Robert Bell. *Advances in collaborative filtering.* Recommender Systems Handbook. Springer US, 2011. 145-186.