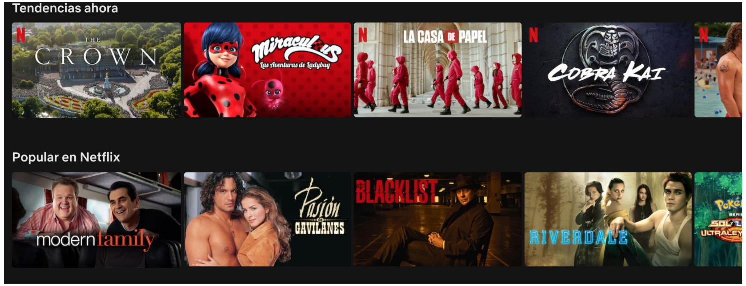
NLP Applications II

Recommender Systems

Introduction

- More and more documents or other kinds of products (*items*) available every day
- Finding the appropriate items might be difficult
- Recommender Systems are applications which provide suggestions for items that might be suitable for a user

Recommender Systems can be



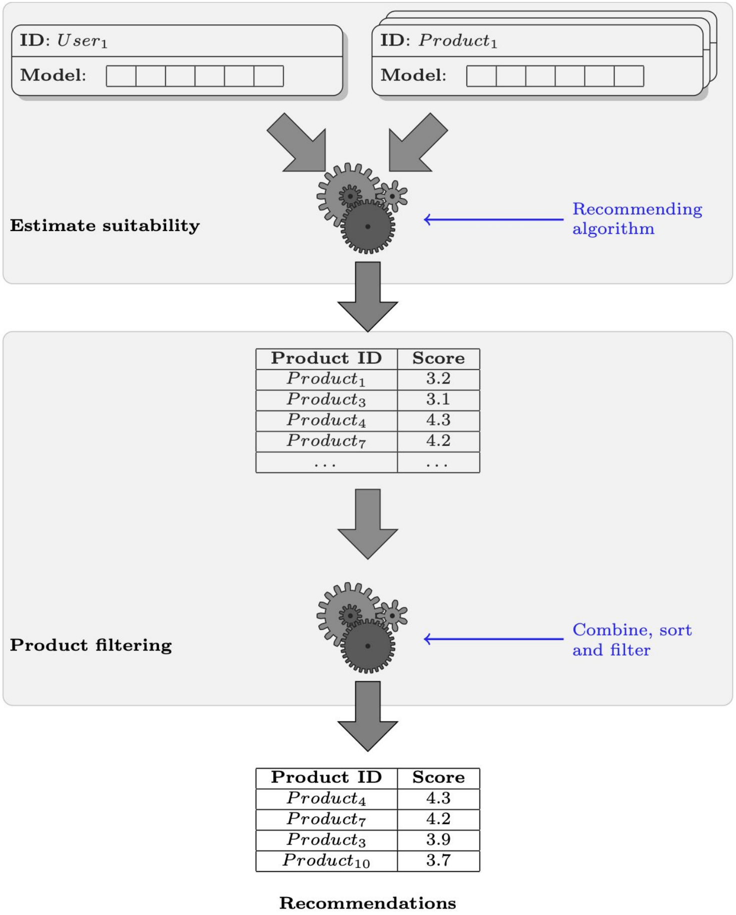
- **Non personalized:**
  - The recommendations are based on popularity
- **Semi-personalized:**
  - The recommendations are filtered according to demographic information (age, location, …)
- **Personalized:**
  - The recommendations are determined considering the interests (profile) of the users
  - In this module, we will focus on this kind of recommender systems

An insight into how personalized recommender systems work

How do we decide which of the movies in the cinema we are going to see this weekend?

- We choose the movie considering the argument, gender, director, actors …
- We listen to the suggestions of our friends, especially those who have similar tastes to ours.
- We chose the movie that is most similar to those films we have seen before and that we liked
- We might even combine some of the other approaches

A generic scheme



Data

Recommender algorithms rely on the rating matrix

Items

Users

5	3			4	2		4	
5	5	2			2	1		4
5	4	3		2	2	1	3	
5	4	?	5	3	3		2	3
2	2	1	2	2	5	6		
	4	5		1	2	3	4	3
		1	2	5	5		4	5
3				4	5	2	3	4
	1	2	1		3	3	5	

The rating matrix is sparse, a lot of data is missing

Data can be

- **Explicit**
    - Most appreciated (easy to interpret), but rare (users do not rate every item)
    - Different kinds
        - Rating (e.g., five-star score in Amazon)
        - Like/dislike
- **Implicit**
    - Records the activity of the users
    - Examples
        - Clicks
        - Purchases
        - Consumed videos
        - Reviews
            - Sentiment analysis, opinion mining
        - ...
    - Harder to interpret, but easier to obtain

Some aspects to be considered about ratings

- Users might rate the items in different way
  - Lenient vs harsh
  - This might affect the estimation of the suitability of the item (rating prediction)
- Normalizing the ratings

  Some approaches

User mean normalization	Z score
$r'_{u,i} = r_{u,i} - \overline{r_u}$	$r'_{u,i} = \frac{r_{u,i} - \overline{r_u}}{\sigma_u}$

A non-exhaustive taxonomy of recommender algorithms

- **Content-based filtering**
  - Make recommendations based on the content
- **Collaborative filtering**
  - Use the “opinions” of other users to make recommendations
  - ***Memory-based***
    - User-User collaborative filtering
    - Item-Item collaborative filtering
  - ***Model-based***
    - Matrix factorization
    - …
- **Hybrid**
  - Combine at least two of the other techniques

Many algorithms rely on similarity

To compare users or items

Most popular approaches:

- **Cosine similarity**

$$sim(i,j) = \frac{\sum_{e \in U_i \cap U_j} r_{i,e} r_{j,e}}{\sqrt{\sum_{e \in U_i} {r_{i,e}}^2 \sqrt{\sum_{e \in U_i} {r_{j,e}}^2}}}$$

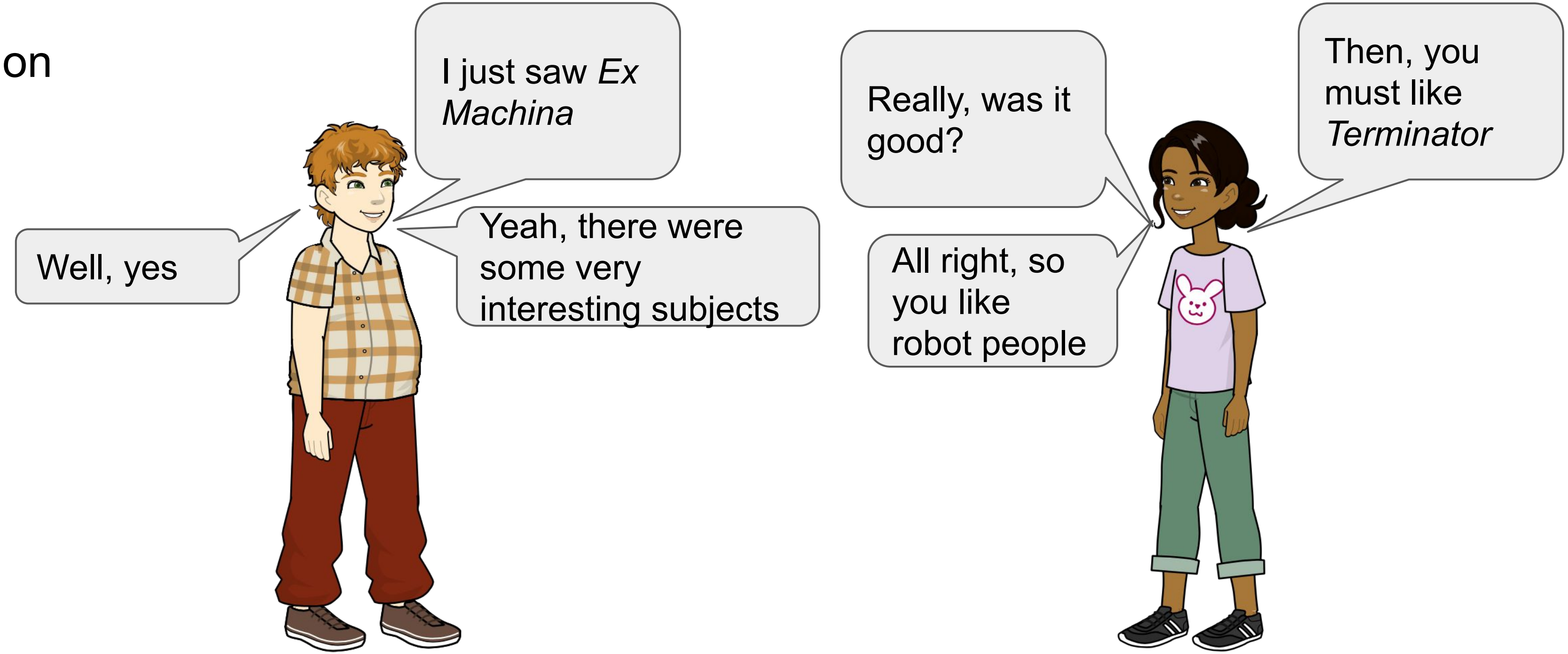
Some implementations only consider ratings in $U_i \cap U_j$

- **Pearson correlation**

$$sim(i,j) = \frac{\sum_{e \in U} \left(r_{i,e} - \bar{r}_i \right) \left(r_{j,e} - \bar{r}_j \right)}{\sqrt{\sum_{e \in U} \left(r_{i,e} - \bar{r}_i \right)^2 \sqrt{\sum_{e \in U} \left(r_{j,e} - \bar{r}_j \right)^2}}}$$

Content-based filtering

Intuition



Source: *Practical Recommender Systems*. Kim Falk. Manning. 2019

Content-based filtering

- A family of algorithms that aim at recommending items which are similar to the preferences of the user.
- Analyse the features/content of the items
  - Facts (objective or certain information)
    - movies: gender, actors, directors, mise en scene data,
    - products: weight, pixel, zoom, ….
    - music: rhythm, tempo, …
    - …
  - Tags (can be more subjective)

Item Modeling

- NLP and Information Retrieval solutions are implemented to this end
- Vector space model
  - $d_i = \{w_{i,1}, w_{i,2}, \dots w_{i,k}\}$
- Different means to represent the item
  - Every dimension has 0/1 (feature applies or does not)
  - Occurrence count (or value)
  - Representations that combine intensity and distinctiveness (e.g., TF-IDF)
- Features that are too common are not useful to filter or discriminate items

TF-IDF

- Defined for Information Retrieval
  - Identify relevant documents
- Statistical which is intended to reflect how important a word is to a document in a collection
- Intuition
  - Term frequency may be significant, it can model the intensity of the term in the document
  - Not all the terms are equally relevant
    - Terms that are used in most documents are not suitable to discriminate or filter documents

$$tf\,idf(w,d) = tf(w,d) \cdot idf(w)$$

$$idf(w) = \log \frac{number\ of\ documents}{number\ of\ documents\ that\ contain\ w}$$

Modeling items using TF-IDF

- Each item is modeled by the vector which contains the TF-IDF score of each feature
- This vector is frequently normalized
- Variants
    - 0/1 frequencies (1 when term occurs above a certain threshold)
    - Logarithmic frequencies (1 + log(tf))
    - Normalized frequency (divided by document length)

User Modeling

- The preferences of the users are inferred considering the items they liked
- **Algorithm**

```
create an empty vector vu
for each item the user liked
    sum the vector vi that represents the item to vu
```

The vector v_i corresponding to those items the user did not like can be subtracted from v_u

Temporal decay (time decay)

Mechanism to model how the interests of the users vary over time

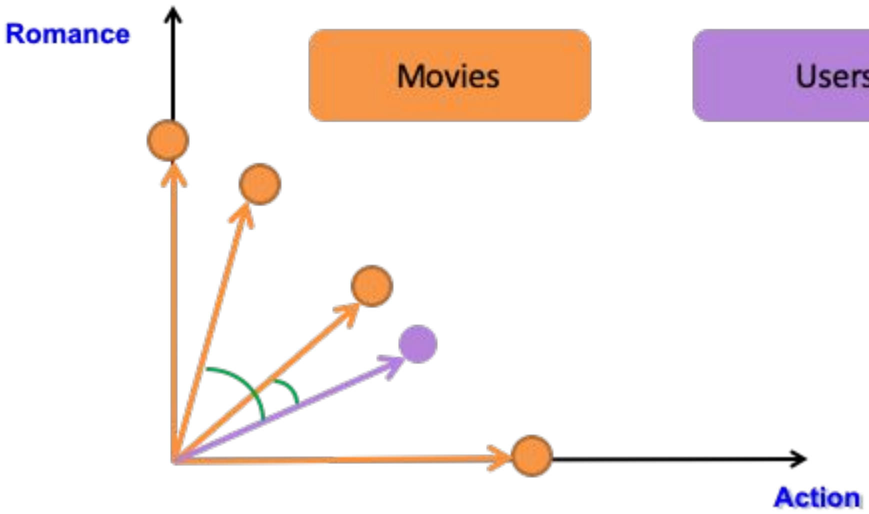
- **Hacker news**

$$r = \frac{\#up\_votes - 1}{(item\_age + 2)^g}$$

- **Reddit**

$$r = \mathrm{sign}(ups - downs) \cdot \log(max(1, |ups - downs|)) + \frac{age}{45000}$$

Determining the suitability of the item for the user



Cosine similarity is used to determine the resemblance between the user profile and the item

Alternative approach

Some implementations model the user as the list of items “consumed” or rated.

This approaches estimate the rating of the item instead

$$r'_{u,i} = \frac{\sum_{j \in I_u} r_{u,j} \cdot sim(i,j)}{\sum_{j \in I_u} sim(i,j)}$$

Performing the recommendations

Algorithm

```
get the list of candidate items I_u //items which have not been consumed by the user u yet
create an empty list of rating R_u
for each item i in I_u
    estimate the adequacy r_ui for the item i
    append the record (i, r_ui) to R_u
sort R_u considering the ratings
return top-n elements in R_u
```

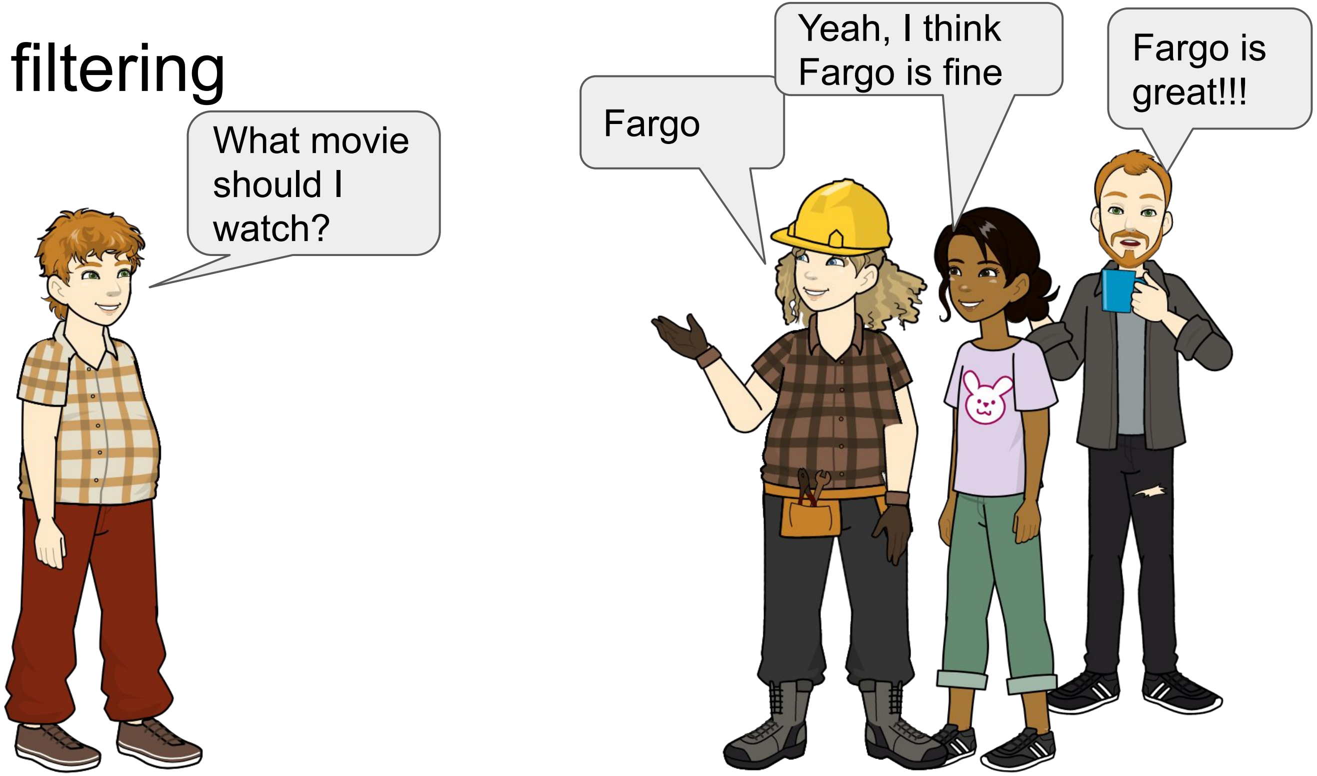
Strengths

- Rely on the metadata (features) of the items
- Easy to implement and to compute
- Understandable profile and recommendations
- Are capable of recommending items, even if they have not been rated by any user yet

Limitations

- Feature engineering can be hard (limited information can affect the recommendation process)
- Tend to overspecialize and recommend items which are very similar to those items the user liked
- Can have problems to handle interdependencies
    - *I like Sandra Bullock in action movies, but Meg Ryan in romantic comedy movies*
    - *I like comedies with violence, and historical documentaries, but not historical comedies or violent documentaries*

Collaborative filtering



Collaborative Filtering algorithms

- Family of algorithm that compute their predictions and recommendations considering the ratings or behaviour of other users in the system
- Base assumptions
  - Our tastes are stable or move in sync with other users' tastes
- Kinds
  - **Memory based** (neighborhood-based):
    - User-User Collaborative Filtering
    - Item-Item Collaborative Filtering
  - **Model based**
    - Matrix factorization
    - ...

User-User Collaborative Filtering

I might like items which similar tastes to mine liked

Procedure

Compute the similarities of the users

Predict the suitability (*rating*) of each candidate item

Sort and filter the rated candidate items

Computing the user similarities

User similarities are considered according to their activity (*ratings)*

Items

Users

5	3			4	2		4	
5	5	2			2	1		4
5	4	3		2	2	1	3	
5	4	?	5	3	3		2	3
2	2	1	2	2	5	6		
	4	5		1	2	3	4	3
		1	2	5	5		4	5
3				4	5	2	3	4
	1	2	1		3	3	5	

Estimating the suitability of the item

Rating predicted based on the opinion of other users (*user neighborhood*)

$$\widehat{r}_{u,i} = \frac{\sum_{j \in N(u,i)} r_{j,i} \cdot sim(u,j)}{\sum_{j \in N(u,i)} sim(u,j)}$$

Using normalized ratings

Users rate using different ranges.

- **User-mean normalization**

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{j \in N(u,i)} \left(r_{j,i} - \bar{r}_j \right) \cdot sim(u,j)}{\sum_{j \in N(u,i)} sim(u,j)}$$

- **Z-score normalization**

$$\hat{r}_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{j \in N(u,i)} \frac{\left(r_{j,i} - \bar{r}_j \right)}{\sigma_j} \cdot sim(u,j)}{\sum_{j \in N(u,i)} sim(u,j)}$$

Size of the neighborhood

How many neighbors should be used?

Usually, 20-100 used

Item-Item Collaborative Filtering

- Similar idea, but based on the similarities of items
- Works better in domains where there are lot more users than items

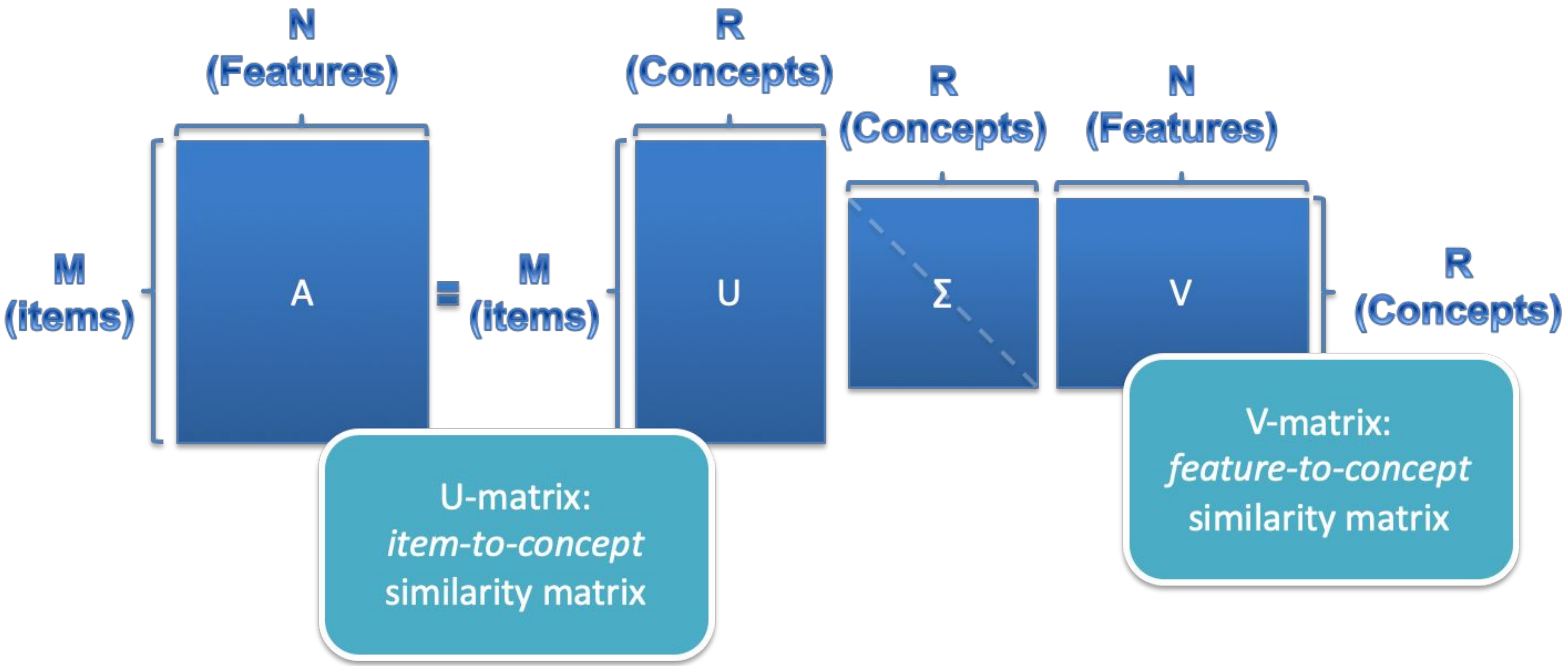
$$\widehat{r}_{u,i} = \frac{\sum_{j \in N(u,i)} r_{u,j} \cdot sim(i,j)}{\sum_{j \in N(u,i)} sim(i,j)}$$

Matrix-Factorization based Collaborative Filtering

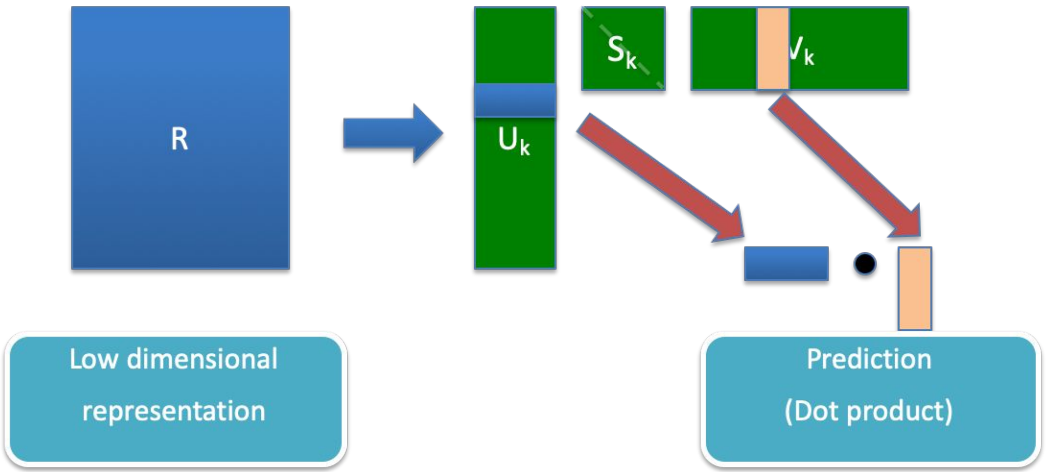
Intuition

The rating matrix can be factorized into two matrices that capture the most salient latent features of the items (and user tastes)

Example: Singular Value Decomposition



SVD for Collaborative Filtering



$$P_{ai} = \sum_{f=1}^{k} u_{af}\sigma_f v_{if}$$

Matrix Factorization in Recommender Systems

SVD too expensive

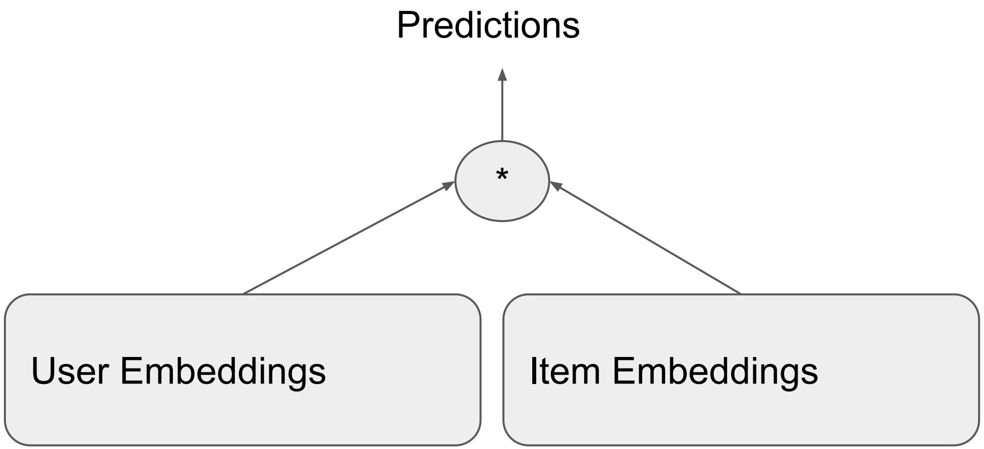
In practice, ML-based approaches

- SVD++
- FunkSVD
- ...

Strengths and limitations

- **Strengths**
  - No metadata required
  - Serendipity
- **Limitations**
  - Cold-start problem

Deep-Learning Based Matrix Factorization



Hybrid Recommender Systems

Combine several recommendation algorithms. For example

Improving Collaborative Filtering Based Recommenders Using Topic Modelling. J. Wilson, S. Chaudhury, B. Lall.
WI-IAT '14: Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01. 2014 Pages 340–346. https://doi.org/10.1109/WI-IAT.2014.54

- Use LDA for topic modelling
- Similarity metrics combines rating overlap and similarity in the latent topic space

Evaluation of recommender systems

Different evaluation approaches

- Offline experiments
- User studies
- Online experiments

Offline experiments

They use available data (*datasets*) to measure whether or not the implemented algorithm is good.

The dataset collects the performance of the users in a context where there is no recommender systems or there was a recommender system which we want to demonstrate is worse

Offline experiments

Approaches

- Accuracy of the predictions
- Adequacy of the recommender decisions
- Adequacy of the rank (Top-n selected items)
- Other aspects
  - Diversity
  - Coverage serendipity
  - ...

Accuracy and adequacy metrics are usually computed per user and, then, averaged

Accuracy metrics

- **Mean Absolute Error (MAE)**

$$MAE = \frac{\sum_{(u,i) \in R} \left| \hat{r}_{u,i} - r_{u,i} \right|}{|R|}$$

- **Mean Squared Error (MSE)**

$$MSE = \frac{\sum_{(u,i) \in R} \left(\hat{r}_{u,i} - r_{u,i} \right)^2}{|R|}$$

- **Root Mean Squared Error (RMSE)**

$$RMSE = \sqrt{\frac{\sum_{(u,i) \in R} \left(\hat{r}_{u,i} - r_{u,i} \right)^2}{|R|}}$$

Caution!!!

- The Netflix prize made RMSE the most popular metric
- Errors can be dominated by irrelevant parts of the item space
- To compare two algorithms, the same dataset must be used
- Recommending items is not just a prediction problem

Recommendation adequacy metrics

- **Precision & Recall** (*hit rate*)

$$precision = \frac{true\ positive}{true\ positive\ +\ false\ positive} \qquad recall = \frac{true\ positive}{true\ positive\ +\ false\ negative}$$

- **Mean Average Precision (MAP)**

$$P@k(u) = \frac{\#\{relevant\ content\ in\ top\ k\ positions\}}{k}$$

$$AP(u) = \frac{\sum_{k=1}^{m} P@k(u)}{m}$$

$$MAP = \frac{\sum_{u \in U} AP(u)}{|U|}$$

Rank adequacy metrics

Errors are penalized by position

- **Mean Reciprocal Rank**

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

rank is the position of the first valid recommendation of the *query i*

- **Normalized Discounted Cumulative Gain**

$$DCG = \sum_{i=1}^{k} \frac{2^{rel(i)} + 1}{\log_2(i + 2)} \qquad nDCG = \frac{DCG}{IDCG}$$

rel is the relevance or gain of the item (can be the rating or the profit of the item)

IDCG is the DCG of the optimal ranking

Limitations of offline experiments

- Restricted to items which have been already rated
  - Good predictions could be actually considered wrong
- Does the recommender really work?
  - Are the users really satisfied by the recommendations?
  - Does the recommender increase sales?

User studies

- Recruited users
- Users interact with the recommender system
- Questionnaires
    - Influence of recommendation algorithm
    - Novelty of the proposals
    - ...

Online Experiments

- Carried out in systems which are in use
- A/B testing
- Part of the traffic (users) are redirected to the system being evaluated
- Record the interaction
- Metrics
    - Click through rate (CTR)
    - ...