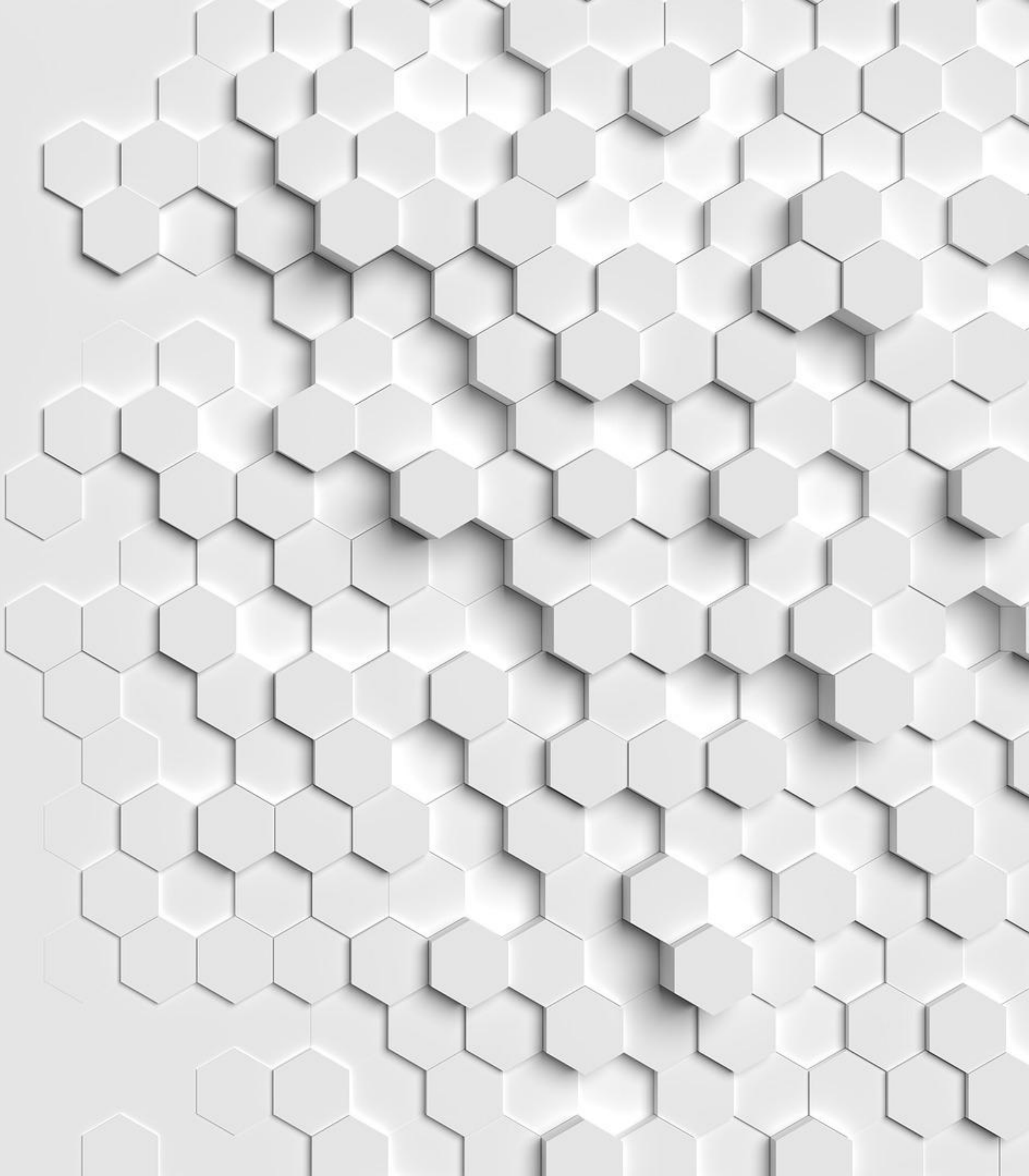
Introduction to recommender systems

Luca Alberto Rizzo



Agenda

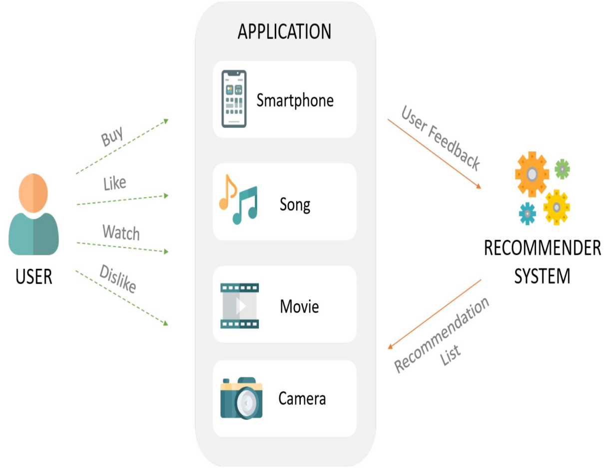
1 Introduction to RecSys

Introduction to RecSys: what is a recommender system?

A friend suggesting you something

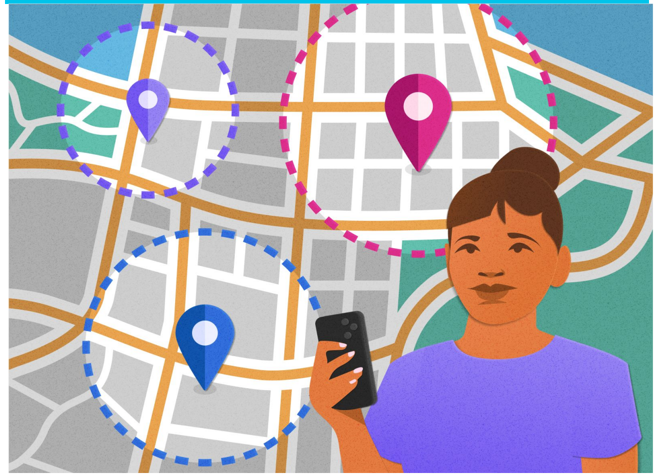
Futurama E18 S4

A complex AI algorithm based on tastes, rating and demographic



Recommendation System Using Autoencoders, Ferreira D., et al.

A marketing campaign algorithm based on your location



Indeed.com

Introduction to RecSys: explicit and implicit feedback

Explicit feedback

- Ratings
- Likes
- Reviews
- …

Build scores ←

Implicit feedback

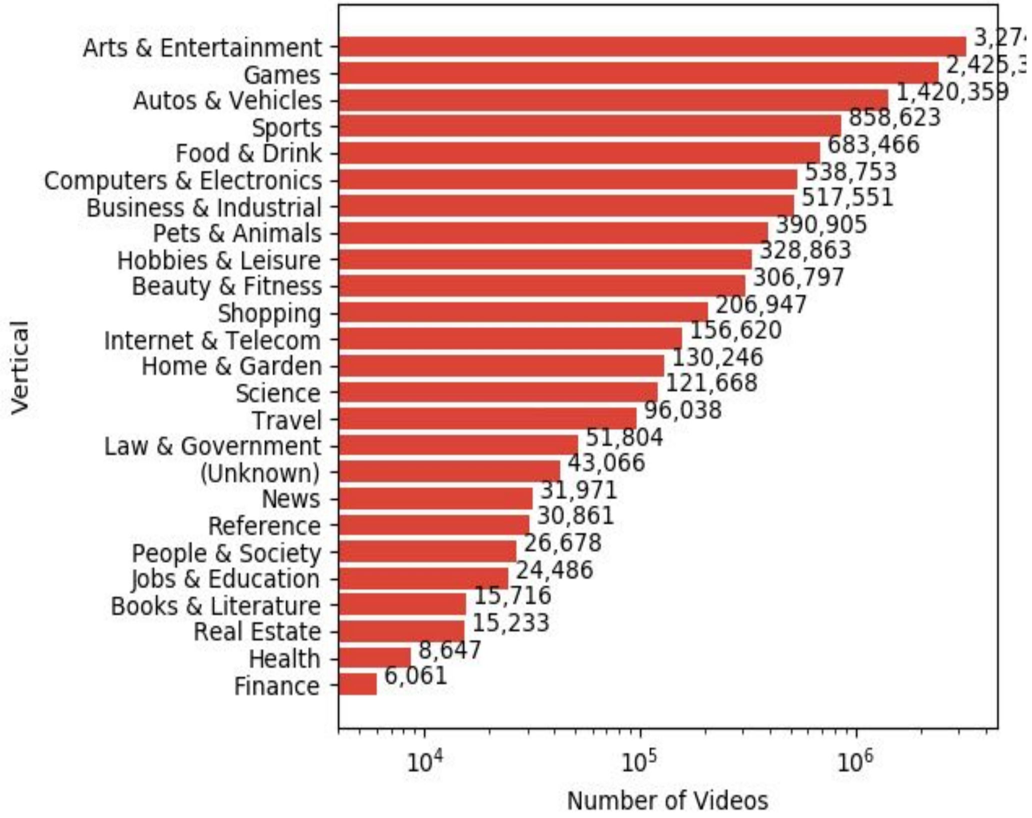
- Purchase history
- Browsing behaviours
- Listening patterns
- …

This lecture focuses on **explicit feedbacks** for simplicity but often **they are not available**

Introduction to RecSys: challenges of modern systems

- **Scalability**: a lot of users, a lot of content
- **Latency**: recommendations have to be delivered in real time
- **Sparsity**: each user interact only with a tiny fraction of items
- **Cold-start**: no information about new users/items

Youtube has approx. 2.1 billion users



[YouTube-8M Segments Dataset](#)

Introduction to RecSys: evaluation metrics

TRAINING METRICS

Mean square rating error

$$\mathrm{MSE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2.$$

Mean square root error

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2.}$$

TESTING METRICS

Precision@k

$$P(k) = \frac{\#\,\text{of top k relevant reco}}{k}$$

P(k=3) = 1/3

Rank	Product	Is recom.	Result
1	product B	1	TP
2	product A	1	FP
3	product E	1	FP
4	product C	1	TP
5	product D	1	TP
6	product G	0	FN
7	product I	0	TN
8	product H	0	TN
9	product F	0	FN

P(k=5) = 3/5

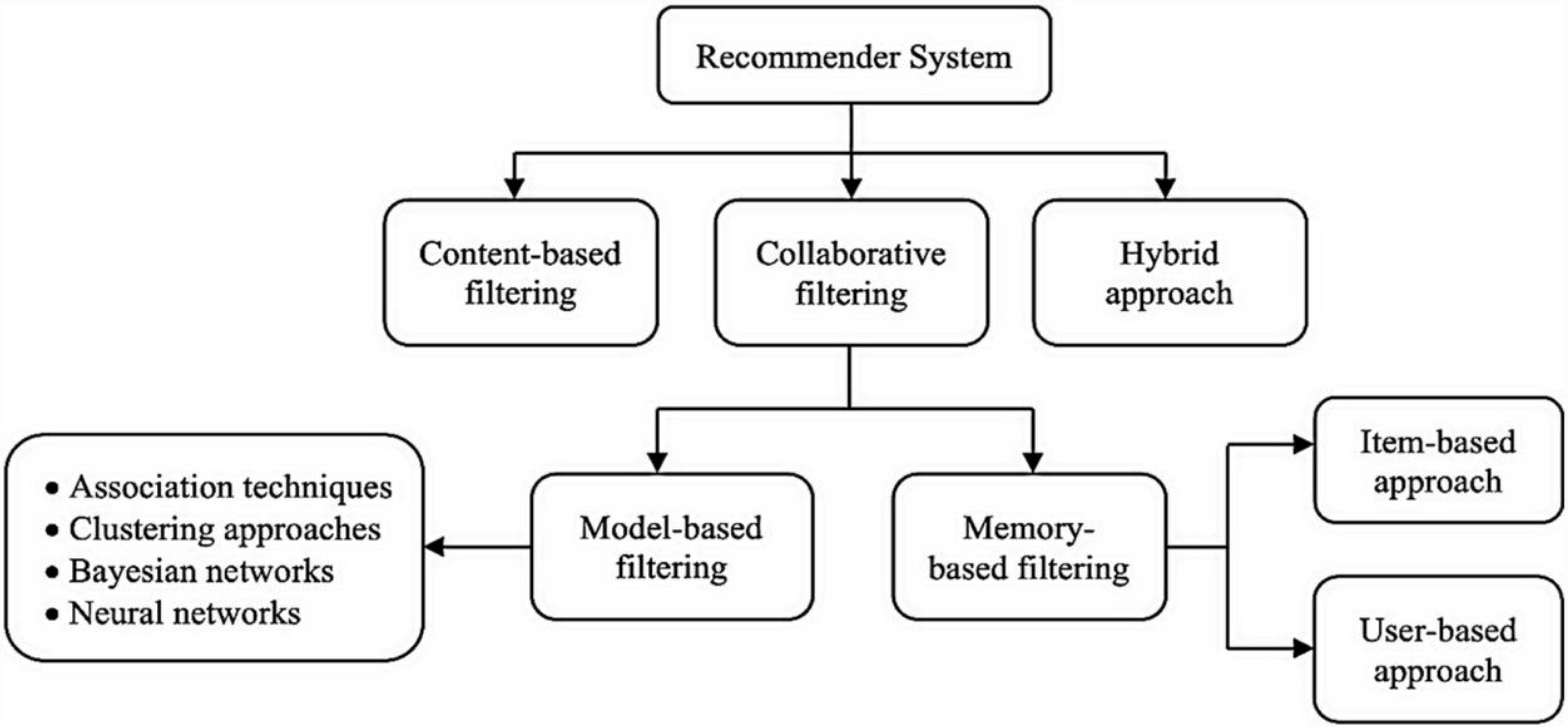
Rank	Product	Is recom.	Result
1	product B	1	TP
2	product A	1	FP
3	product E	1	FP
4	product C	1	TP
5	product D	1	TP
6	product G	0	FN
7	product I	0	TN
8	product H	0	TN
9	product F	0	FN

Average precision

$$\mathrm{AP@N} = \frac{1}{m} \sum_{k=1}^{N} (P(k) \text{ if } k^{th} \text{ item was relevant})$$

Other metrics can be defined to study diversity, novelty, etc. A good place to start is Neptune.ai blog

Introduction to RecSys: taxonomy



From: [A systematic review and research perspective on recommender systems]

2 Content/user based RecSys

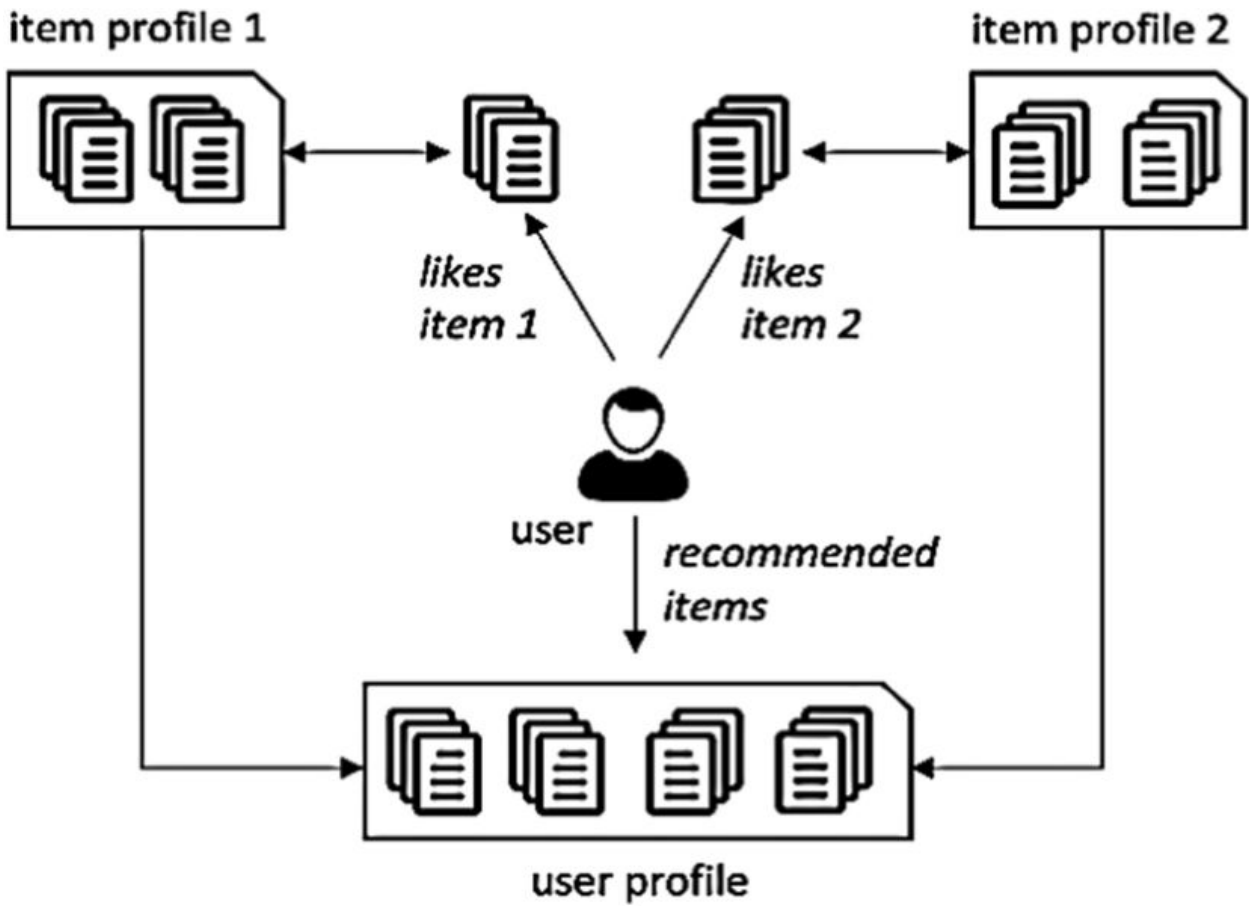
Content-based RecSys: similarity

ALGORITHM

1. Collect user **consumptions/likes** (list of items)
2. Collect **features (e.g. genre, length, …)** of consumed items
3. Compute **similarity** between new item and consumed ones
4. Recommends items with **higher similarity**

Cosine similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$



From: [A systematic review and research perspective on recommender systems]

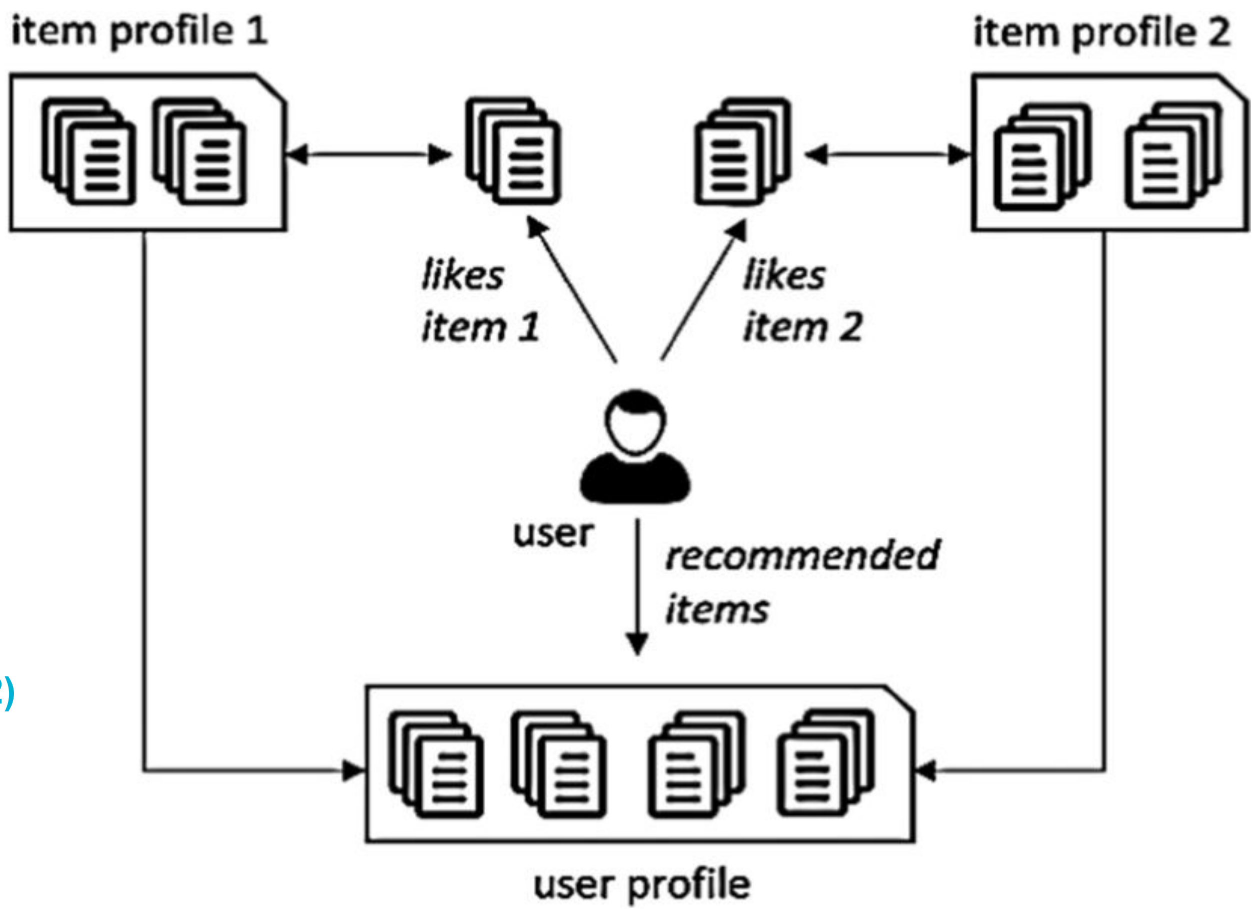
Content-based RecSys: similarity

ADVANTAGES

1. **“Quickly” gives “good” results**
2. Ensure **user privacy** (no need for user features)

DRAWBACKS

1. **Not scalable** since each pair should be computed **O(N^2)**
2. Requires **in-depth knowledge of item features**



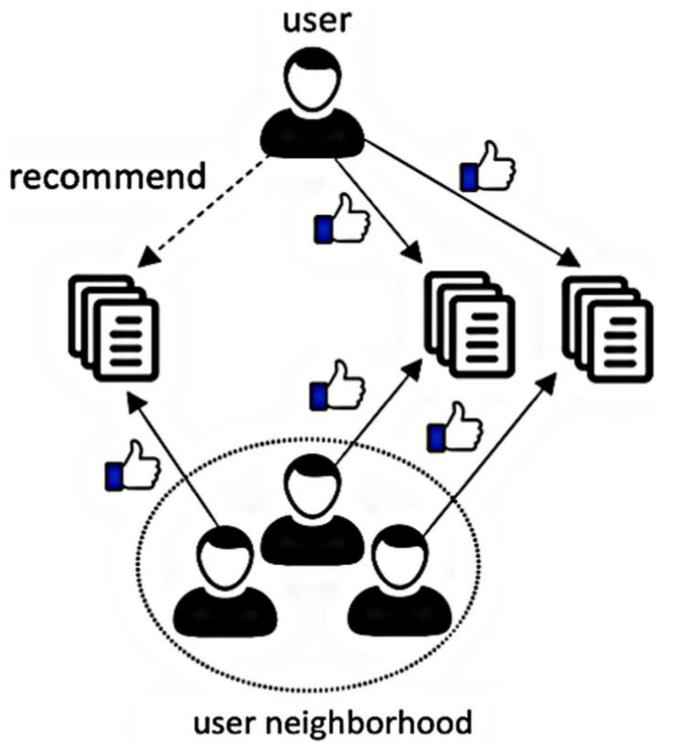
From: [A systematic review and research perspective on recommender systems]

3 Collaborative filtering

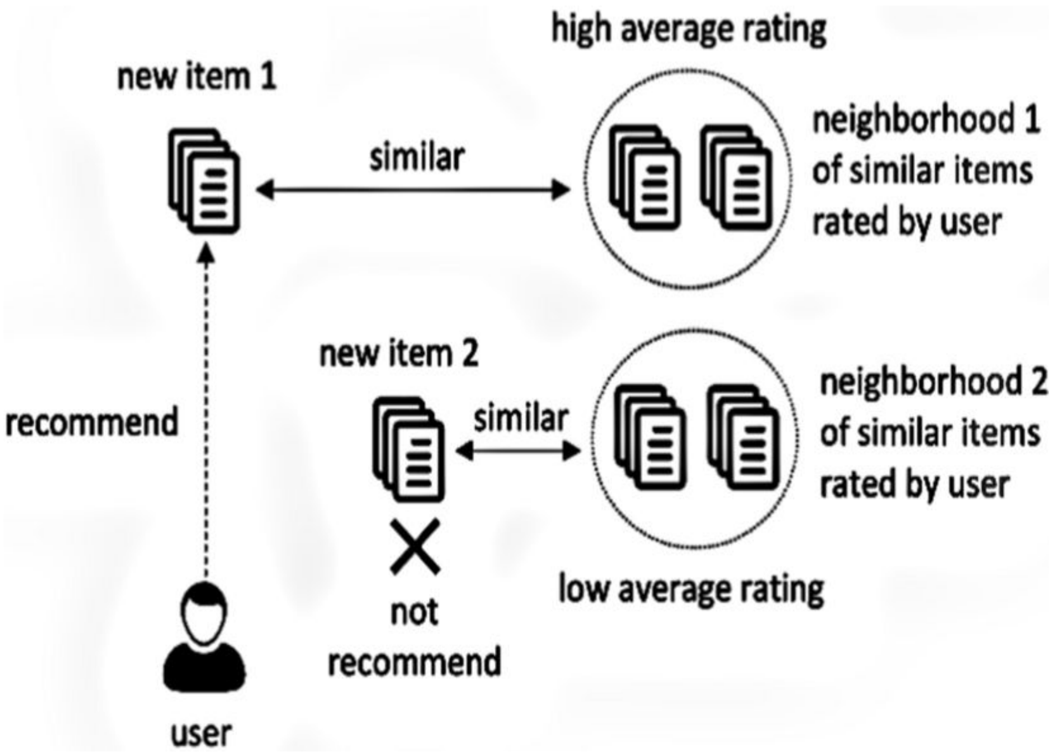
Collaborative filtering: introduction

Recommends highly rated items belonging to **similarity neighbourhoods** (items or users)

User-based collaborative filtering



Item-based collaborative filtering



From: [A systematic review and research perspective on recommender systems]

Collaborative filtering: KNN-inspired algorithm

1. Compute **user feature neighborhoods** by minimizing:

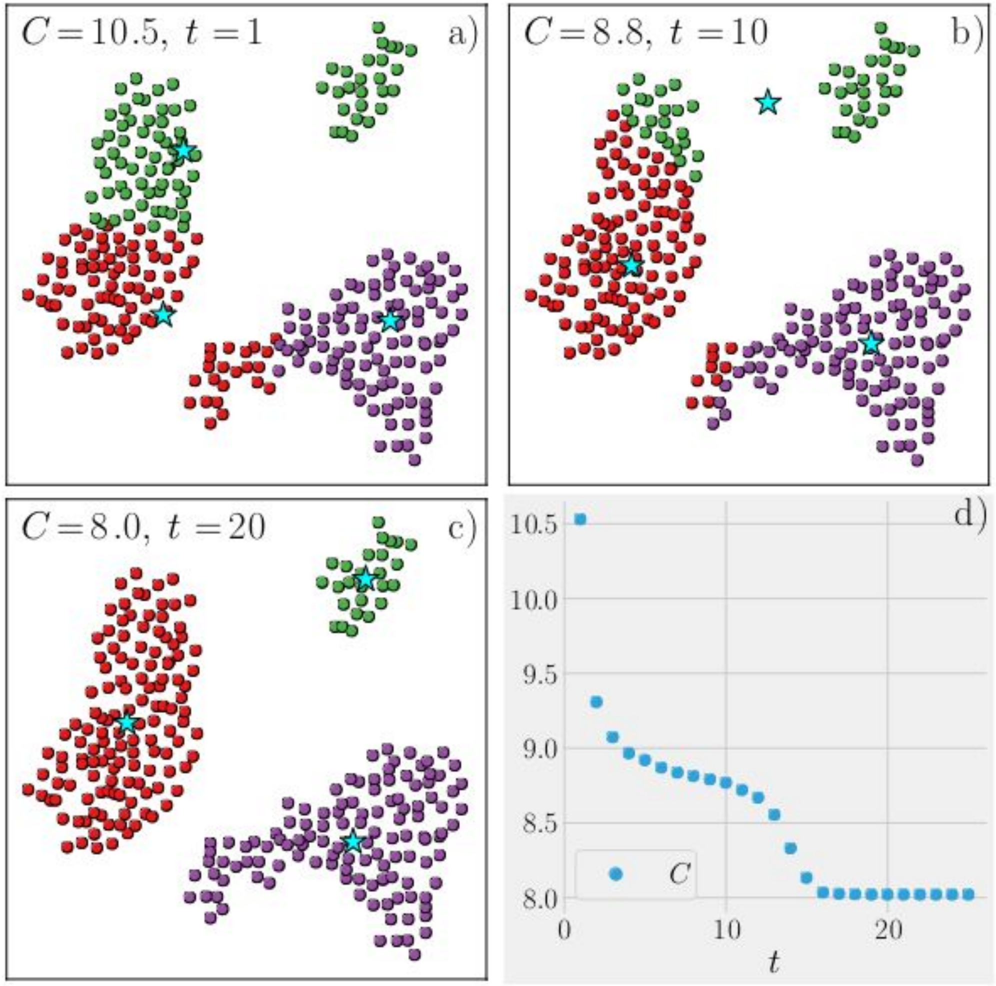
$$\mathcal{C}(\{x, \boldsymbol{\mu}\}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)^2,$$

2. **Predicted rating** for each items is assigned as:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

i.e. as a weighted average for neighbors ratings

3. Recommend items with **highest ratings in the neighborhood**



From A high-bias, low-variance introduction to Machine Learning for physicists

Collaborative filtering: Netflix Prize

"Open competition for best collaborative filtering algorithm to predict user ratings, only based on **previous ratings**"

Approx 1.5 M quadruplets
<user, movie, date of grade, grade>

Improve **of RSME of 10%** compared to Cinematch Netflix algorithm

Started in 2006, it took more **than 3 years** for a team to achieve this result!

Winner's solution was never adopted,
but **3rd team's solution revoluzioned RecSys**

Netflix prize winners, from left: Yehuda Koren, Martin Chabbert, Martin Piotte, Michael Jahrer, Andreas Toscher, Chris Volinsky and Robert Bell.

Collaborative filtering: matrix factorization

Matrix factorization aims to represent users and items in a **lower dimensional latent space**

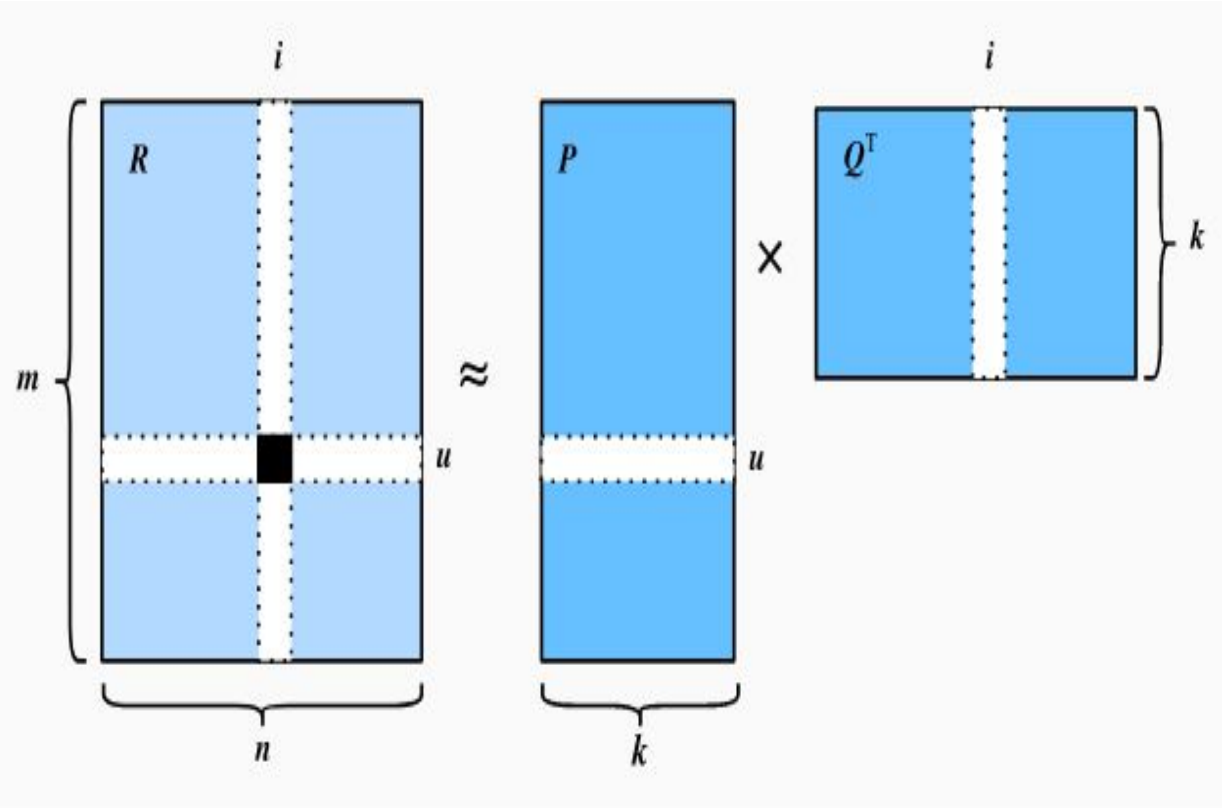
SVD algorithm computes the **predicted ratings** as:

$$\hat{\mathbf{R}}_{ui} = \mathbf{p}_u \mathbf{q}_i^\top + b_u + b_i$$

where b_u and b_i are user and item bias terms

Model HP are obtained by minimizing:

$$\underset{\mathbf{P},\mathbf{Q},b}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{K}} \|\mathbf{R}_{ui} - \hat{\mathbf{R}}_{ui}\|^2 + \lambda(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + b_u^2 + b_i^2)$$



Source: Dive into deep Learning

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. 2008

Collaborative filtering: Neural collaborative filtering

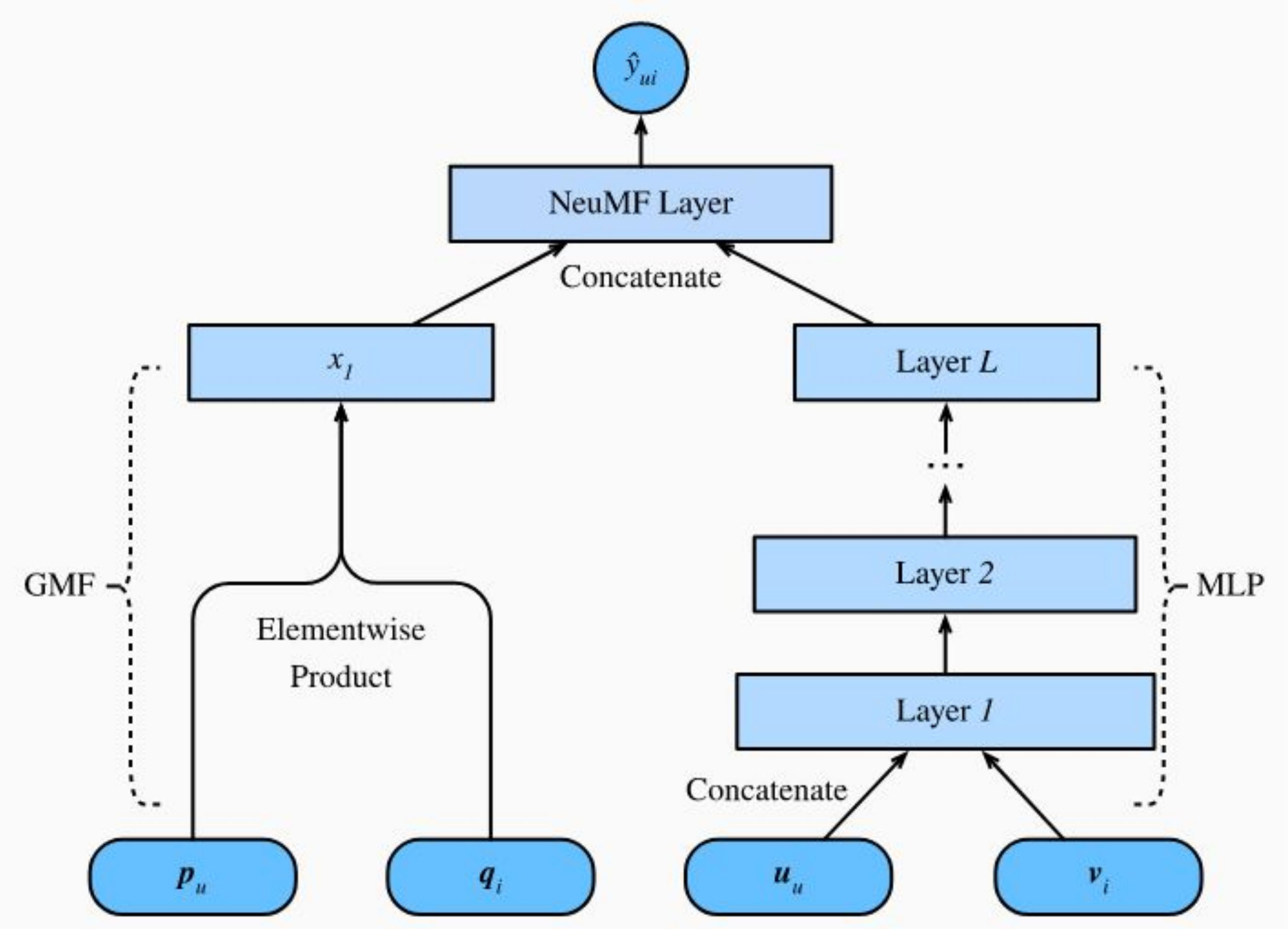
Matrix factorization can be combined with Deep Learning to learn lower feature representations **more efficiently**

Neural version of MF: the input is the [Hamard product](#) **of user and item latent factors**

$$\mathbf{x} = \mathbf{p}_u \odot \mathbf{q}_i$$
$$\hat{y}_{ui} = \alpha(\mathbf{h}^\top \mathbf{x}),$$

with a MLP subnetwork

$$\begin{aligned} z^{(1)} &= \phi_1(\mathbf{U}_u, \mathbf{V}_i) = [\mathbf{U}_u, \mathbf{V}_i] \\ \phi^{(2)}(z^{(1)}) &= \alpha^1(\mathbf{W}^{(2)} z^{(1)} + b^{(2)}) \\ &\ldots \\ \phi^{(L)}(z^{(L-1)}) &= \alpha^L(\mathbf{W}^{(L)} z^{(L-1)} + b^{(L)})) \\ \hat{y}_{ui} &= \alpha(\mathbf{h}^\top \phi^L(z^{(L-1)})) \end{aligned}$$



Source: Dive into deep Learning

Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction.

4 Movielens Dataset

Movielens dataset: introduction

MovieLens is a non commercial web-based RS
that recommends movies for its users to watch

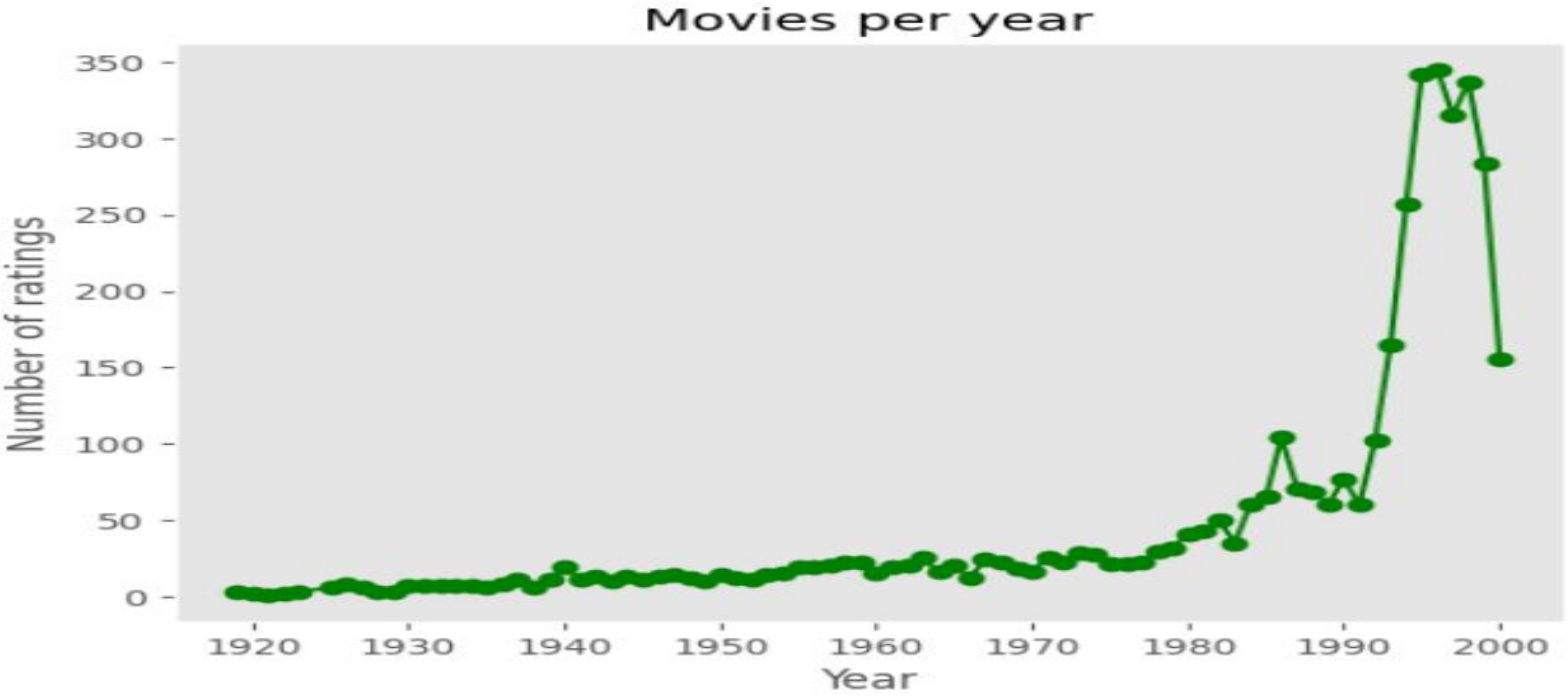
The GroupLens Research data at the University of Minnesota created the MovieLens dataset with:

- 26M ratings
- 750k tag applications
- 45k movies
- 270k users

Download and further explanation can be found here

Considered the **standard for research RS**

Movielens dataset: some simple analysis

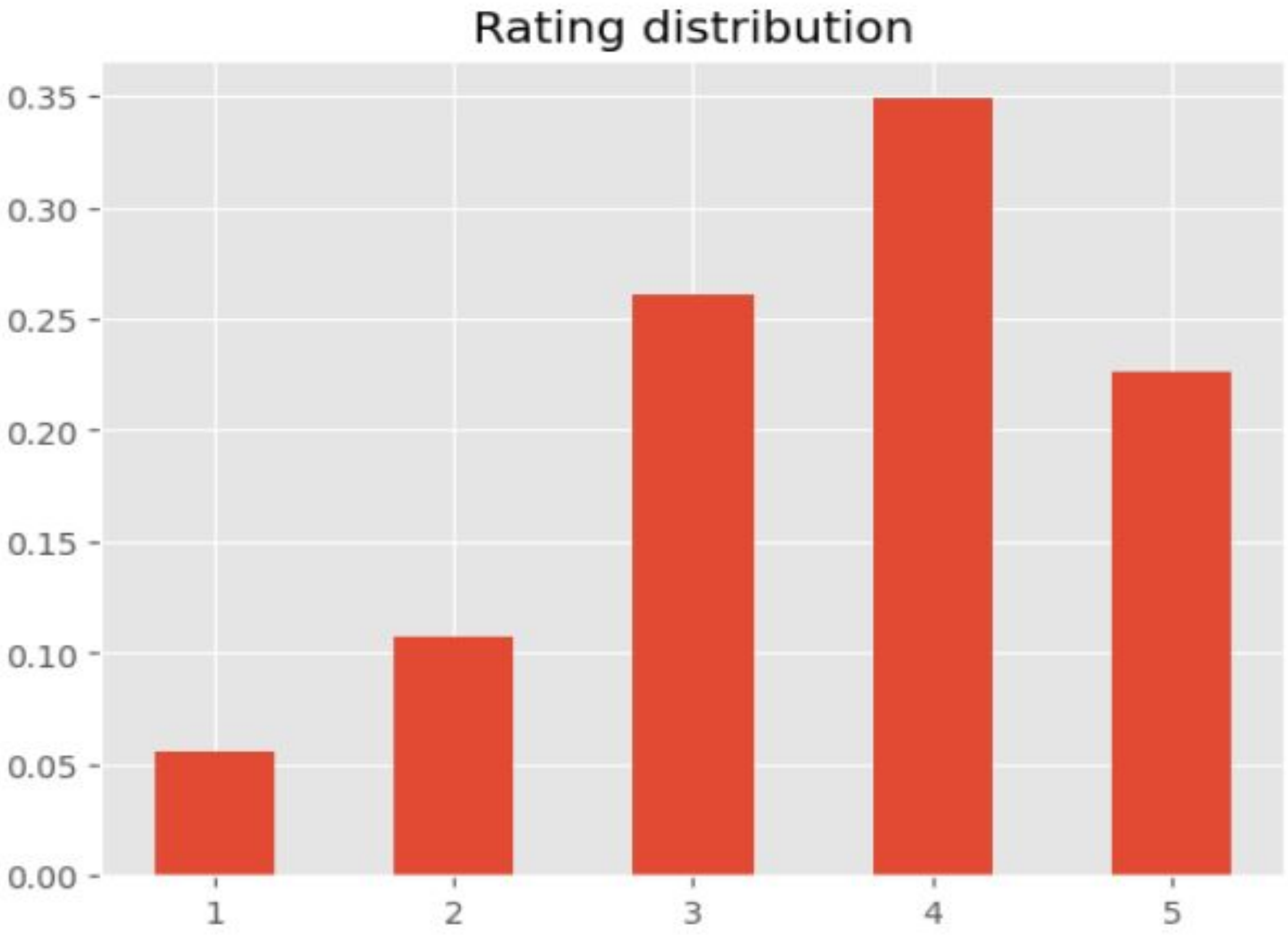


Prevalence of movies late **‘90 and early 2000**

Movielens dataset: some simple analysis

We will use a 10k and 1M version of MovieLens

	user_id	item_id	rating	timestamp
0	0	0	3	881250949
1	1	1	3	891717742
2	2	2	1	878887116
3	3	3	2	880606923
4	4	4	1	886397596
...	...	...	...	...

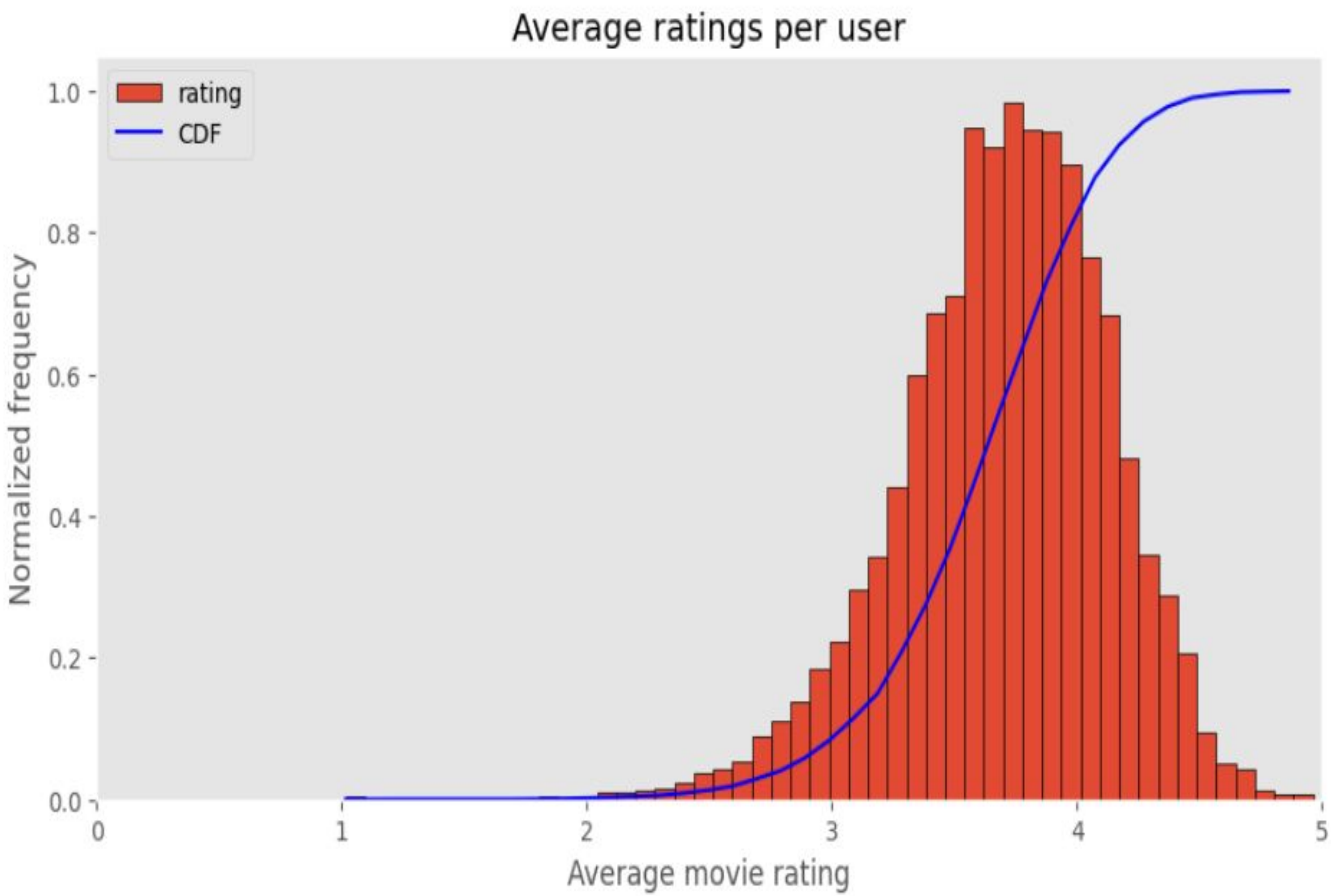


Ratings **not equally distributed**

Movielens dataset: some simple analysis

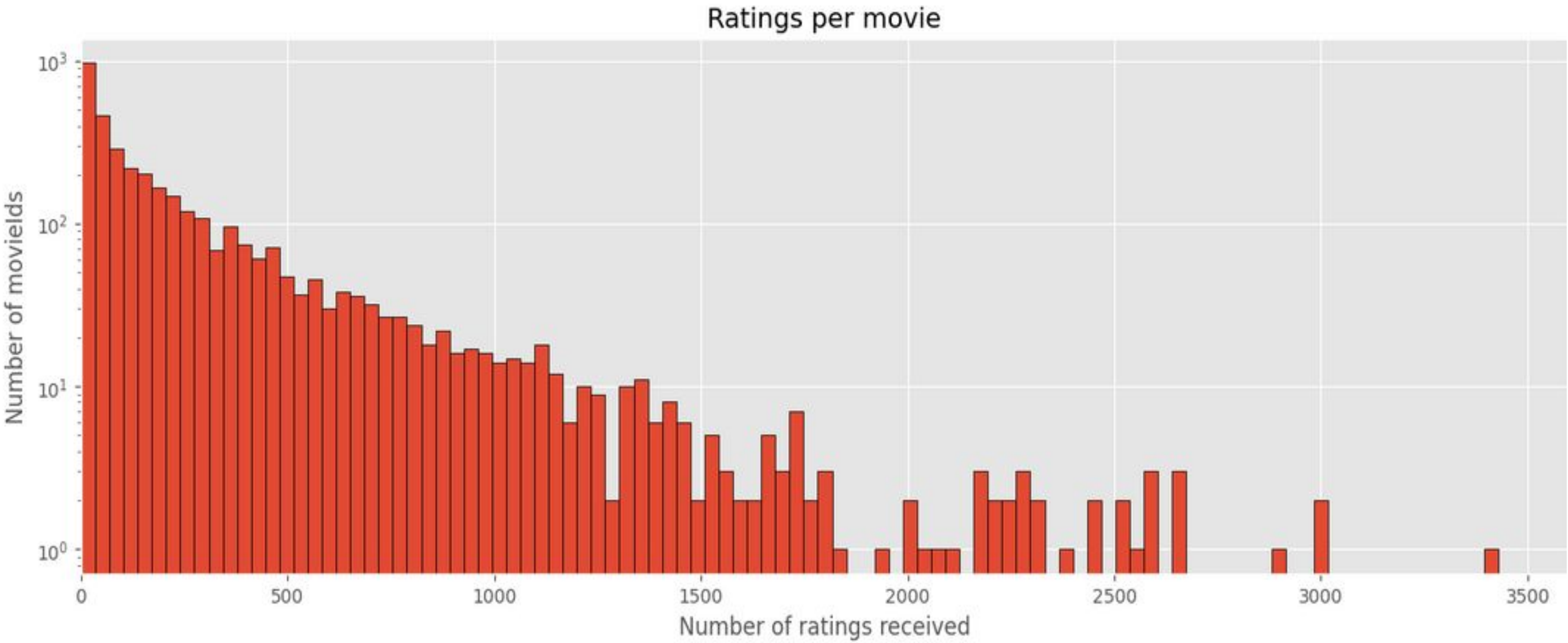
We will use a 10k and 1M version of MovieLens

	user_id	item_id	rating	timestamp
0	0	0	3	881250949
1	1	1	3	891717742
2	2	2	1	878887116
3	3	3	2	880606923
4	4	4	1	886397596
...	...	...	...	...



Users are **positively biased**

Movielens dataset: some simple analysis

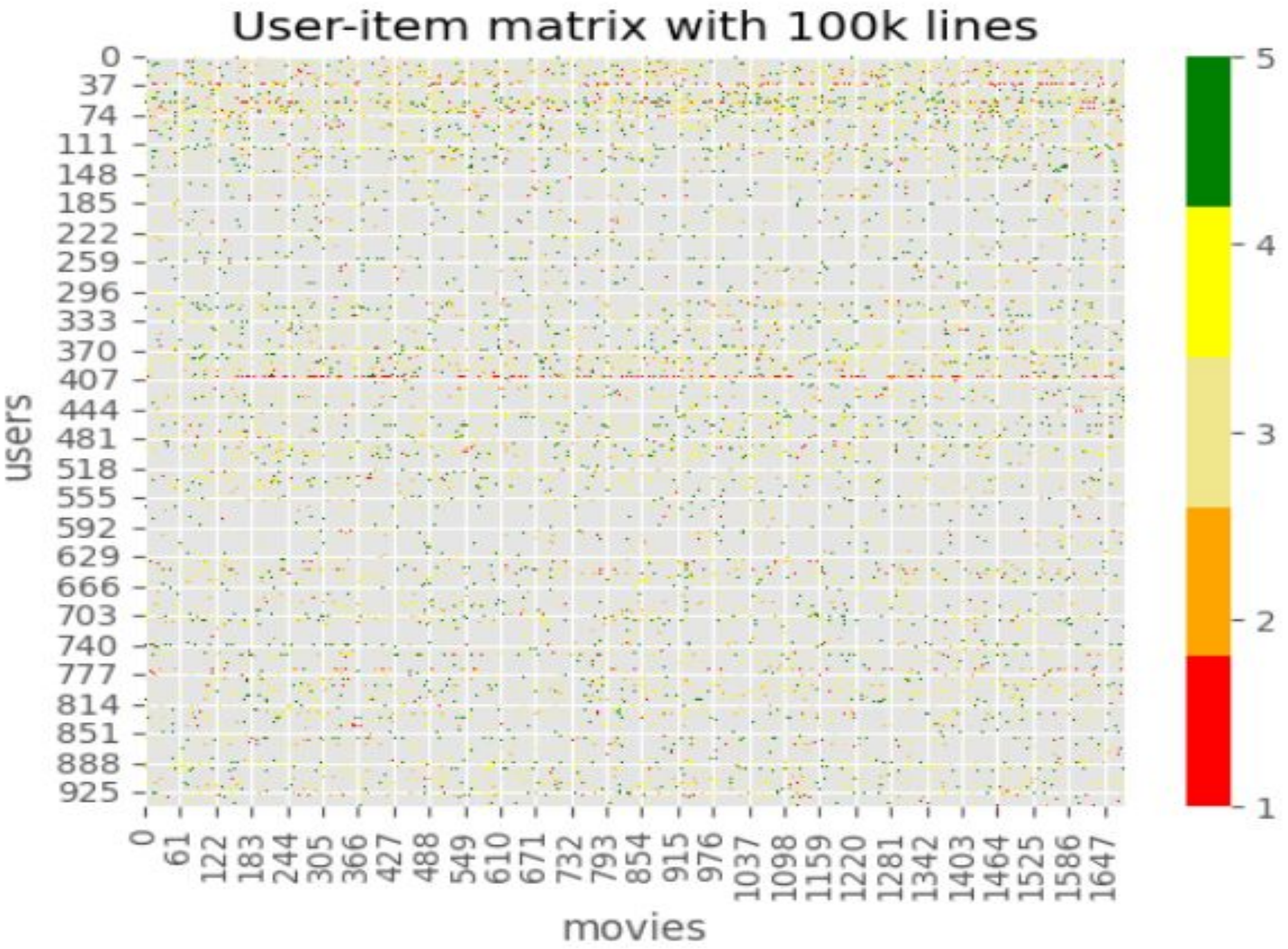


Some movies **are much more rated than others**

Movielens dataset: some simple analysis

Matrix sparsity

$$S(M_{u,i}) = \frac{N_{\emptyset}}{N_i \times N_u} \simeq 0.006$$



Very sparse matrix at 100k → **gets much worse**

5 Surprise library

Surprise library: introduction

SurPRISE stands for **Simple Python Recommendation System Engine:**

- easy to **use and install library** with emphasis on documentation
- provide r**eady-to-use prediction algorithms** such as neighborhood methods and matrix factorization, and many others
- make **easy to handle datasets** such as MovieLens
- provides **a large choice of metrics** to evaluate RecSys
- has **build-in CV tools** for easy comparison among models

```
pip install numpy
pip install scikit-surprise
```

simple pip installation

[Surprise library documentation]

Surprise library: import datasets

1. **Load the movielens-100k** dataset (download it if needed)

```python
from surprise import Dataset
data = Dataset.load_builtin('ml-100k')
```

2. By default dataset is **"surprise.dataset" object** optimized for training

```python
data
```

```
<surprise.dataset.DatasetAutoFolds at 0x7fe85a5e5270>
```

3. Csv data **are saved in local "surprise_data" folder,** ready for further analysis

```python
data_df = pd.read_csv('/home/user/.surprise_data/ml-100k/ml-100k/u.data',
                      sep = '\t',
                      names = ['user_id','item_id','rating','timestamp'])
data_df.head()
```

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

Surprise library: test with 100k MovieLens

KNN (Basic)

```
from surprise import KNNBasic

algo = KNNBasic()
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).
```

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9740	0.9828	0.9810	0.9752	0.9801	0.9786	0.0034
MAE (testset)	0.7669	0.7796	0.7745	0.7694	0.7749	0.7731	0.0045
Fit time	0.49	0.45	0.40	0.45	0.46	0.45	0.03
Test time	2.75	2.89	2.61	2.81	2.65	2.74	0.10

Matrix factorization

```
from surprise import NMF

algo = NMF()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm NMF on 5 split(s).
```

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9617	0.9703	0.9592	0.9636	0.9592	0.9628	0.0041
MAE (testset)	0.7587	0.7622	0.7534	0.7579	0.7506	0.7566	0.0041
Fit time	1.68	1.63	1.76	1.88	1.84	1.76	0.09
Test time	0.08	0.09	0.27	0.08	0.12	0.13	0.07

Similar results with 100k MovieLens (5 folds cross validation)

Surprise library: test with 1M MovieLens

KNN (Basic)

```python
from surprise import KNNBasic

algo = KNNBasic()
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9241  0.9216  0.9234  0.9224  0.9228  0.9229  0.0009
MAE (testset)     0.7286  0.7260  0.7279  0.7270  0.7276  0.7274  0.0009
Fit time          28.32   28.61   27.62   26.42   27.03   27.60   0.80
Test time         108.82  106.59  99.35   101.73  104.46  104.19  3.37
```

Matrix factorization

```python
from surprise import NMF

algo = NMF()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm NMF on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9183  0.9165  0.9178  0.9181  0.9144  0.9170  0.0014
MAE (testset)     0.7259  0.7230  0.7255  0.7246  0.7223  0.7242  0.0014
Fit time          16.26   16.11   18.23   16.79   16.12   16.70   0.80
Test time         1.26    2.07    1.40    1.53    1.74    1.60    0.29
```

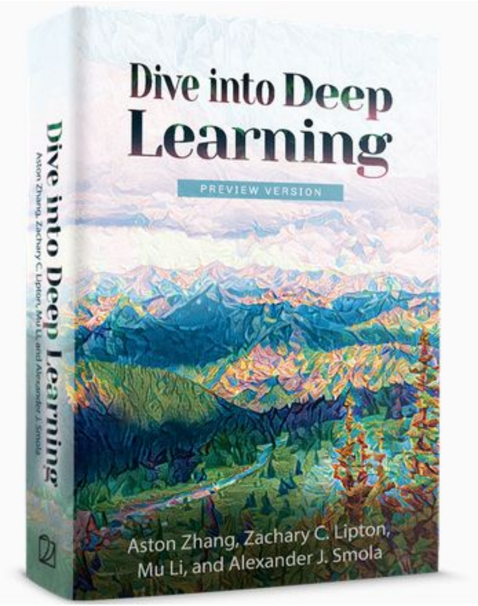
Similar results but much test time almost 100 times higher with 1M MovieLens (5 folds cross validation)

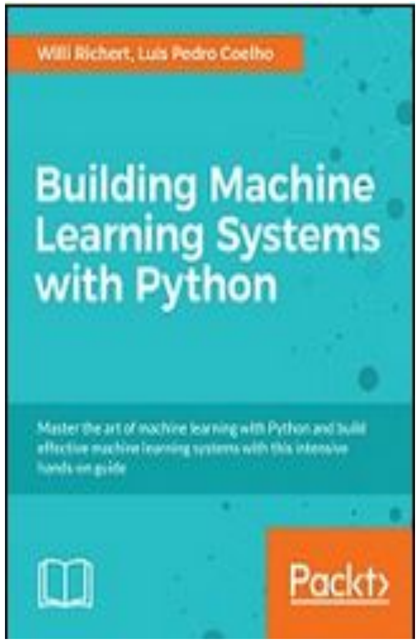
6 Conclusions

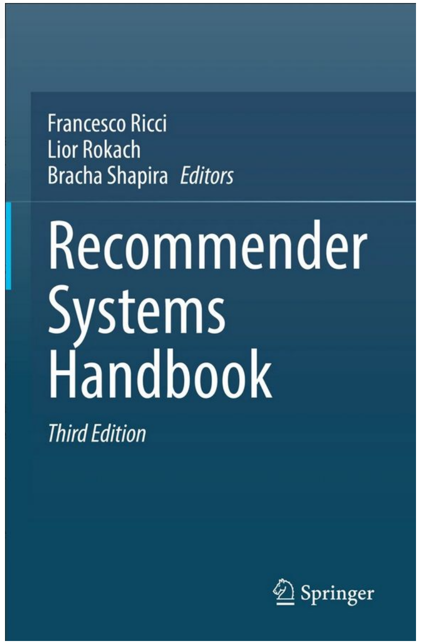
Conclusions

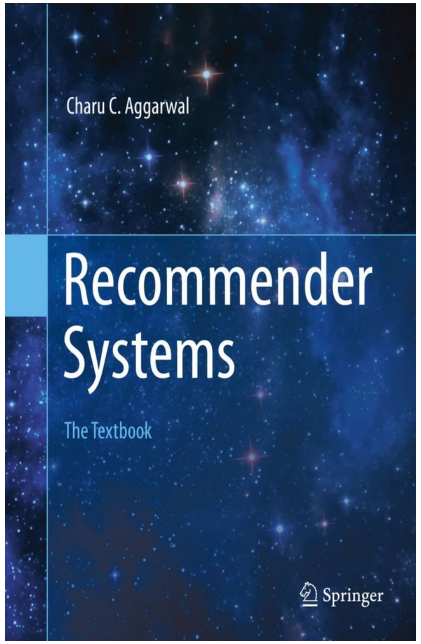
- Modern RecSys **are powerful** but suffer of some problems, such as **sparsity, cold start …**
- Content/used based RecSys are an easy but useful solution **with “little data”**
- **Collaborative filtering** is based on the idea of using other user’s information
- **KNN based algorithm** are useful but do not scale well
- **MF is the “golden standard”** of RecSys as it scales well and it is designed for sparse data
- NN based implementation of MF are available but they not be useful with “little data”

Further material









Thank you for you attention

introduction-reco-lagos

Luca Alberto Rizzo

luca.alb.rizzo@gmail.com