MULTIMEDIA RETRIEVAL

Week9

Semester 1, 2025

Recommender Systems

- Background
- Recommendation algorithms
  - Collaborative filtering
    - User based
    - Model based
      - Matrix factorization
  - Content-based
    - Product, document, image, video, audio
  - Learning based
- Context Aware Recommendation
- Evaluation

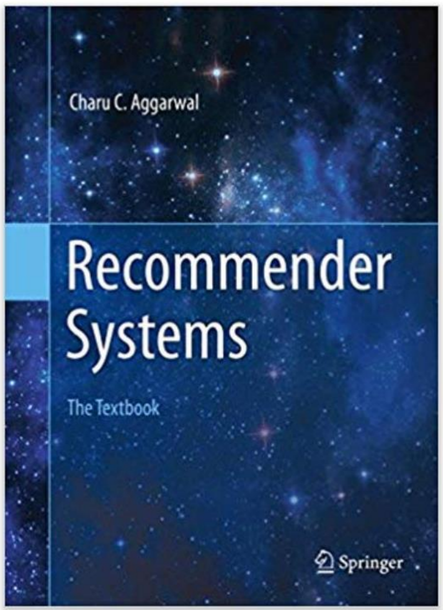
Recommendation is everywhere

Recommender Systems: The Textbook 1st ed. 2016 Edition

by Charu C. Aggarwal (Author)

9 customer reviews

Look inside



ISBN-13: 978-3319296579
ISBN-10: 3319296574
Why is ISBN important?

eTextbook	Hardcover	Paperback	Other Sellers
$59.50	$62.62	$65.49 - $69.99	See all 4 versions

Buy new

In Stock.

Ships from and sold by Amazon.com.

Coupon Save an extra $5.91 when you apply this coupon. Details

This item ships to **Australia**. **Get it by Friday, March 22 - Wednesday, March 27** Choose this date at checkout.
Learn more

More Buying Choices

27 New from $62.62 | 19 Used from $71.68

ISBN-13: 978-3319296579
ISBN-10: 3319296574
Why is ISBN important?

Have one to sell? Sell on Amazon

Add to List

Share

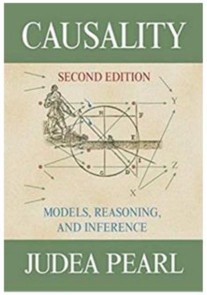
More Buying Choices

27 New from $62.62 | 19 Used from $71.68

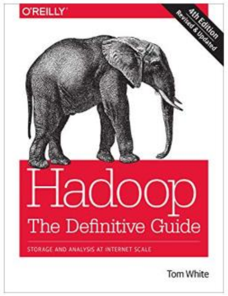
This book comprehensively covers the topic of recommender systems, which provide personalized rec
of products or services to users based on their previous searches or purchases. Recommender systen
have been adapted to diverse applications including query log mining, social networking, news recomn
and computational advertising. This book synthesizes both fundamental and advanced topics of a rese
has now reached maturity. The chapters of this book are organized into three categories:

‹ Read more

Customers who bought this item also bought



Causality: Models, Reasoning and Inference
› Judea Pearl
38
Hardcover
$52.99



Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale
› Tom White
67
#1 Best Seller in Parallel Computer Programming
Paperback
$36.06

You may also like

Jack & Jones
JAMIE - Polo shirt - orange
£21.00
Free delivery & returns

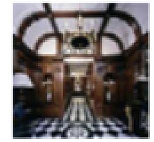
ALTERNATIVE PRODUCTS

Beko Washing Machine
Code: WMB81431LW
£269.99

Zanussi Washing Machine
Code: ZWH6130P
£269.99

Blomberg Washing Machine
Code: WNF6221
£299.99

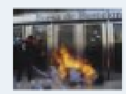
Related hotels...



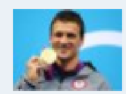
Hotel 41
1,170 Reviews
London, England

Show Prices

Read | Commented | Recommended



Germany Just Rejected The Idea That The European Bailout Fund Would Buy Spanish Debt ×



There Is Almost No Gold In The Olympic Gold Medal ×

You may also like

(109)

(53)

(33)

MOST POPULAR | RECOMMENDED

How to Break NRA's Grip on Politics: Michael R. Bloomberg

Growth in U.S. Slows as Consumers Restrain Spending

Recommendation is everywhere

- eCommerce
  - Amazon, eBay, …
- Social
  - Facebook, LinkedIn, …
    - Friends, groups, jobs
- Media
  - Youtube, Netflix, Spotify, …
  - News
  - Advertisement
- Others
  - MOOC, tourism, …

Benefits of RecSys

- For customers or users
  - Find relevant things
  - Narrow down the set of choices
  - Help explore the space of options
  - Discover new things
  - …

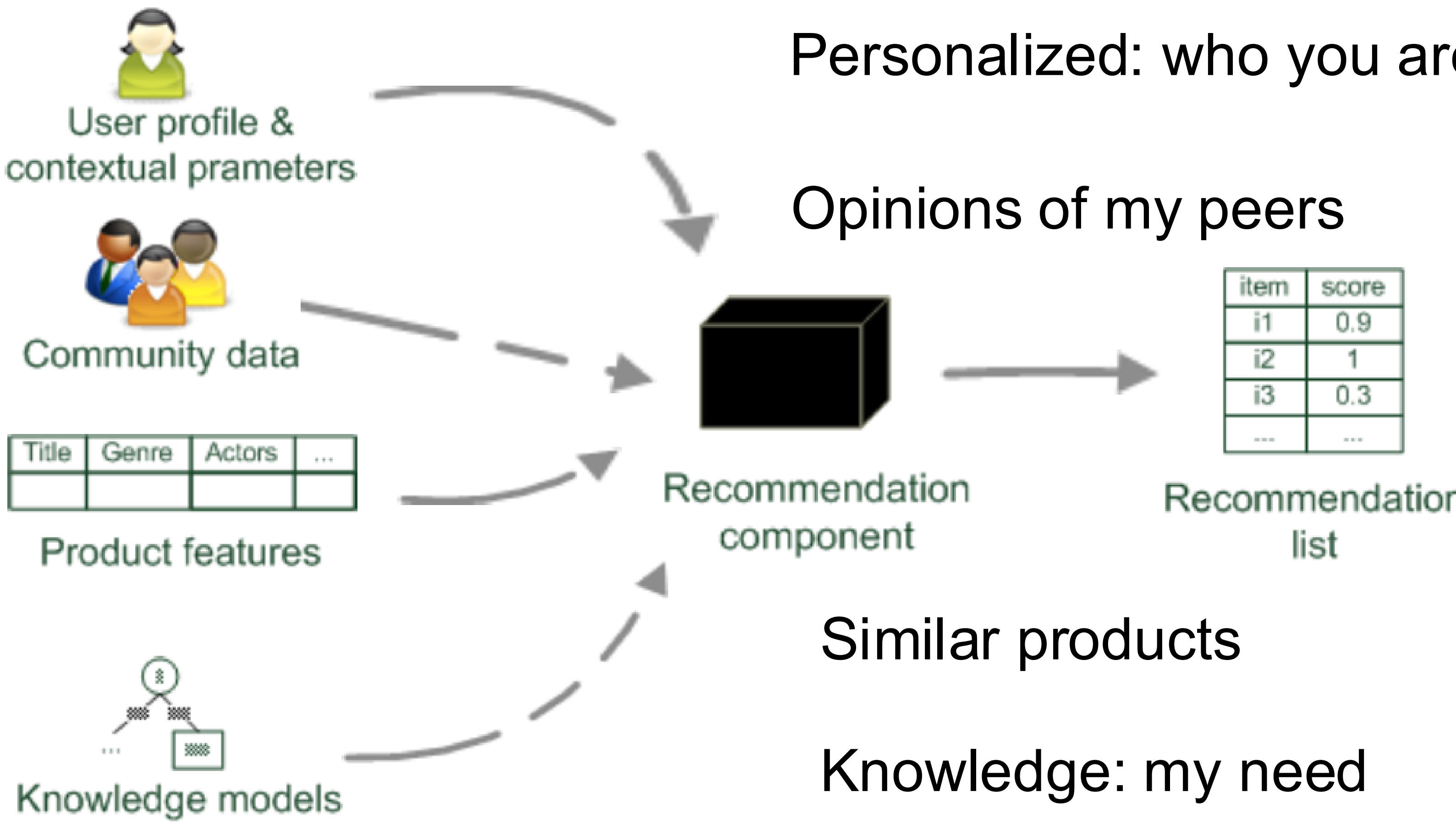
- For providers or vendors
  - Additional and probably unique personalized or customized service
  - Increase trust and customer loyalty
  - Increase sales (30% - 70%), click through rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

Problem Statement

- Input
  - User model and profile (e.g., ratings, preferences, and other meta data)
  - Items (with or without attributes)

- Goal
  - Recommend items to potential users
    - Relevance score in terms of various criteria (e.g., context)
  - Obtain missing values between users and items
    - Netflix: 100K movies, 10M users, 1B ratings

Paradigms of RecSys



Personalized: who you are

Opinions of my peers

Similar products

Knowledge: my need

Collaborative Filtering

- Problem
  - Input: users provide ratings for some items (explicitly or implicitly)
  - Output: produce missing ratings between users and items

- Idea
  - Users having similar ratings have similar interests or preferences
  - Recommend items rated highly by similar users, but not rated by the current user
    - Nappy vs Beer

- The most practical and prominent approach!

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, Using collaborative filtering to weave an information tapestry, Communications of the ACM, 35(12): 61-70, Dec 1992.

User based Nearest-neighbour CF

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

- Select to most similar users (peers) to the active user over a target item
- Aggregate (e.g. average) the ratings of the peers

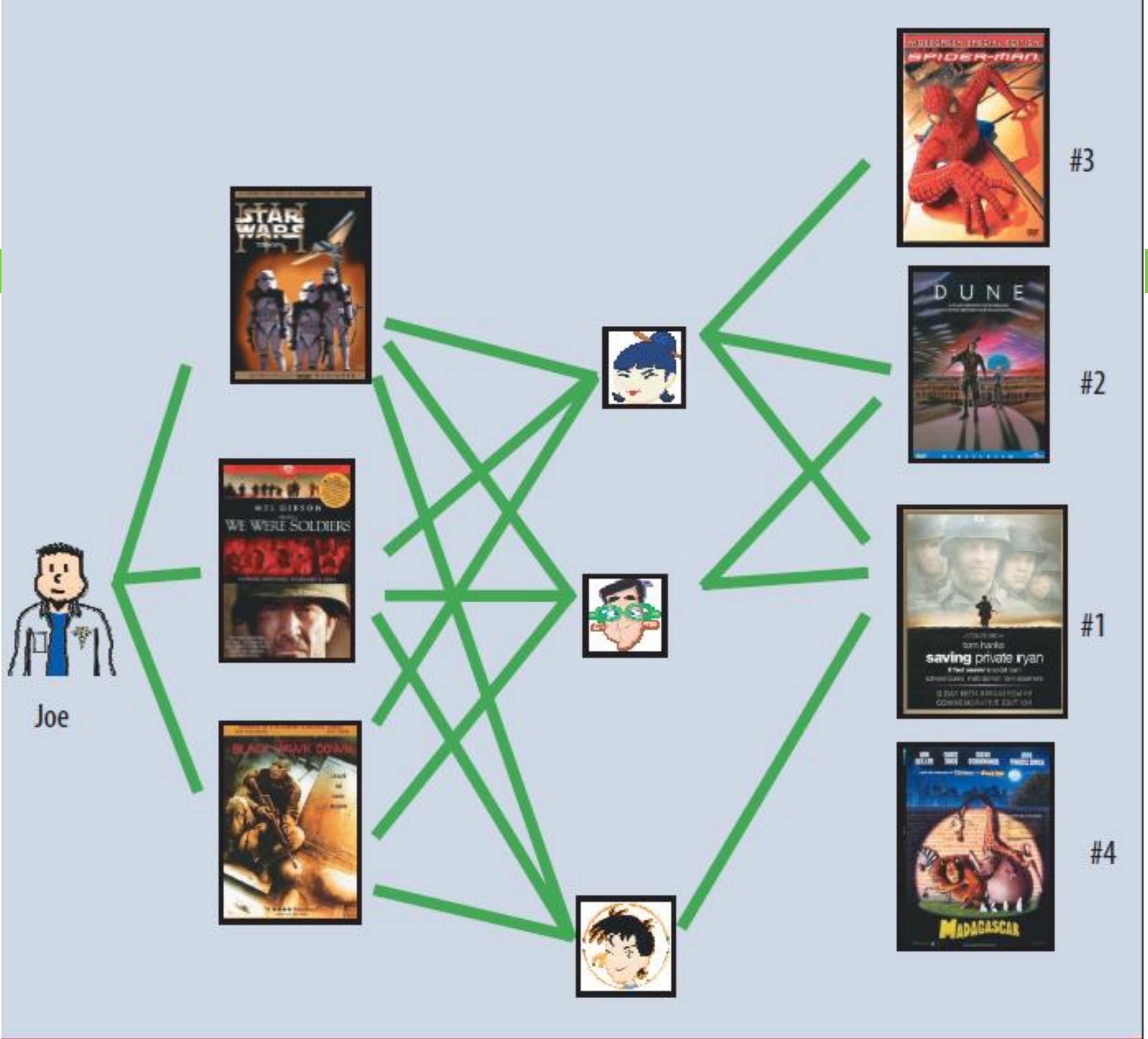


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

User based Nearest-neighbour CF

- Given an "active user" Alice and an item *i* not yet seen by Alice

- To estimate Alice's rating (i.e. interest) over this item *i*
  - Find a set of users (peers) who liked the same items as Alice in the past AND who have rated *i*
  - Aggregate the ratings of the peers for producing the ratings of Alice over *i*
  - Perform this for all the items Alice has not seen and identify the best rated items

User based Nearest-neighbour CF

- Questions
  - How to decide peers
    - Who and how many
  - How to aggregate

Similarity metric: correlation

- The similarity of two users(' history)
- Consider only the items which have been rated by both of them

Pearson correlation: $\text{sim}(X, Y) = \dfrac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$

Similarity metric

- Alice vs User 1
  - $\overline{r_{Alice}} = \overline{r_a} = 4$
  - $\overline{r_{User1}} = \overline{r_b} = 2.4$

$$sim(a,b) = \frac{\sum_{p=1}^{n}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p=1}^{n}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p=1}^{n}(r_{b,p} - \overline{r_b})^2}}$$

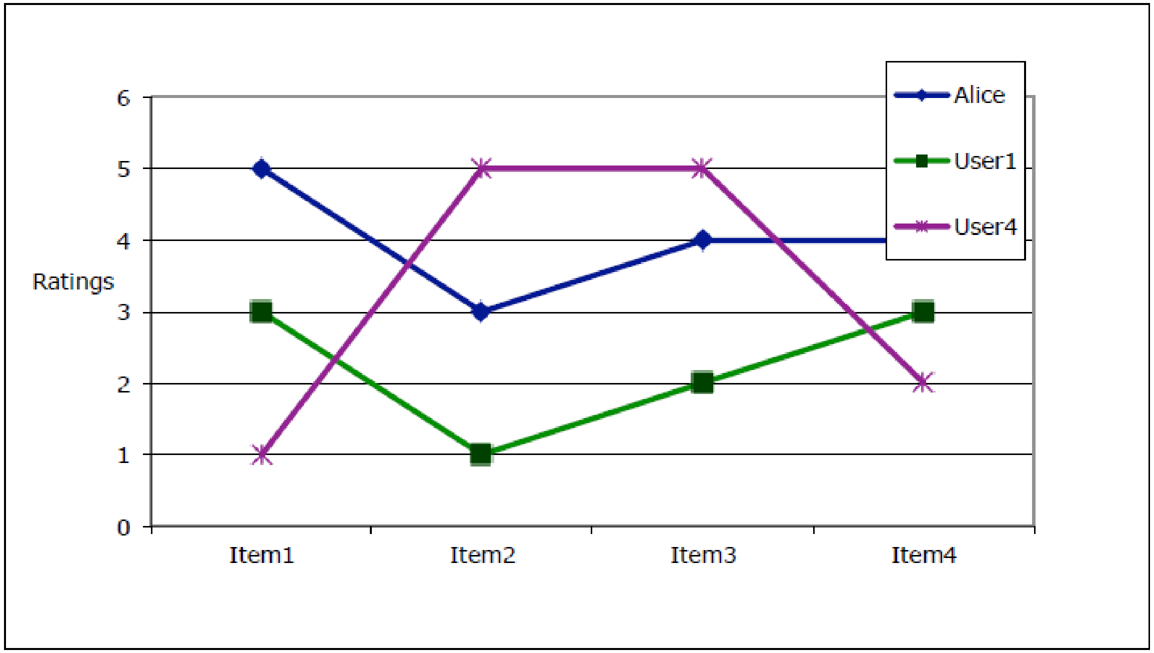
$$\frac{(5-\overline{r_a})\times(3-\overline{r_b})+(3-\overline{r_a})\times(1-\overline{r_b})\cdots+(4-\overline{r_a})\times(3-\overline{r_b})}{\sqrt{(5-\overline{r_a})^2+(3-\overline{r_a})^2}\ \ldots\sqrt{(3-\overline{r_b})^2+(1-\overline{r_b})^2}\ \ldots} = 0.85$$

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85
sim = 0,70
sim = -0,79

Pearson Correlation

- Works well in usual domains, compared with alternatives
  - Cosine similarity



Recommendation

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

- Users: *a* and *b*
- Ratings
  - $\overline{r_a}$: the average rating of *a*
  - $r_{b,p}$: the rating on item *p* from user *b*
- Similarity weight
  - *sim(a,b)*: correlation between *a* and *b*

Recommendation

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Prediction for Alice's rating on Item5 based on the rating of her nearest neighbors (User1 and User 2):

$$4 + \frac{0.85 \times (3 - 2.4) + 0.7 \times (5 - 3.8)}{0.85 + 0.7} = 4.87$$

Recommendation

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	5
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Improving the metrics

- Not all neighbor ratings might be equally "valuable"
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - Possible solution: Give more weight to items that have a higher variance
- Value of number of co-rated items
  - Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
- Case amplification
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- Neighborhood selection
  - Use similarity threshold or fixed number of neighbors
  - More recently, social recommenders use social relations (e.g. friendship) to select “similar” users rather than the full set of users

Rating Prediction

- Predict a rating, $p_{a,i}$, for each item i, for active user, a, by using the n selected neighbor users, $u \in \{1,2,\ldots n\}$.
- To account for users different ratings levels, base predictions on *differences* from a user's *average* rating.
- Weight users' ratings contribution by their similarity to the active user.

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} w_{a,u}}$$

Significance Weighting

- Do not to trust correlations based on very few co-rated items
- Include *significance weights*, $s_{a,u}$, based on number of co-rated items, *m*.

$$w_{a,u} = s_{a,u} \, \mathrm{sim(a, u)}$$

$$s_{a,u} = \left\{ \begin{array}{l} 1 \text{ if } m > 50 \\ \dfrac{m}{50} \text{ if } m \le 50 \end{array} \right\}$$

Memory based (user based) vs Model based (item based)

- User-based CF is said to be "memory-based"
  - the rating matrix is directly used to find “similar” users to make predictions
  - does not scale for most real-world scenarios (unless we know something about the users, other than the previous purchases)
  - large e-commerce sites (Amazon, NeSlix) have tens of millions of customers and millions of items (but they are just a few companies, while many companies are interested in recommending but have cold-start problem)
- Model-based CF approaches
  - based on an offline pre-processing or "model-learning" phase
  - at runtime, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - large variety of techniques used
  - model-building and updating can be computationally expensive

Item-based CF

- Basic idea
  - Item-based (model-based) CF exploits relationships between items first, instead of between users

- Relationship between items can be computed offline

B. Sarwar et al., Item-based collaborative filtering recommendation algorithms, WWW 2001.

Item-based CF

- Basic idea
  - User the similarity between items to make predictions
  - However, we need to know something about the items (e.g., descriptions)
- Example
  - Look for items that are similar to Item 5 (as for rating)

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Item5

Cosine Similarity

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Adjusted cosine similarity

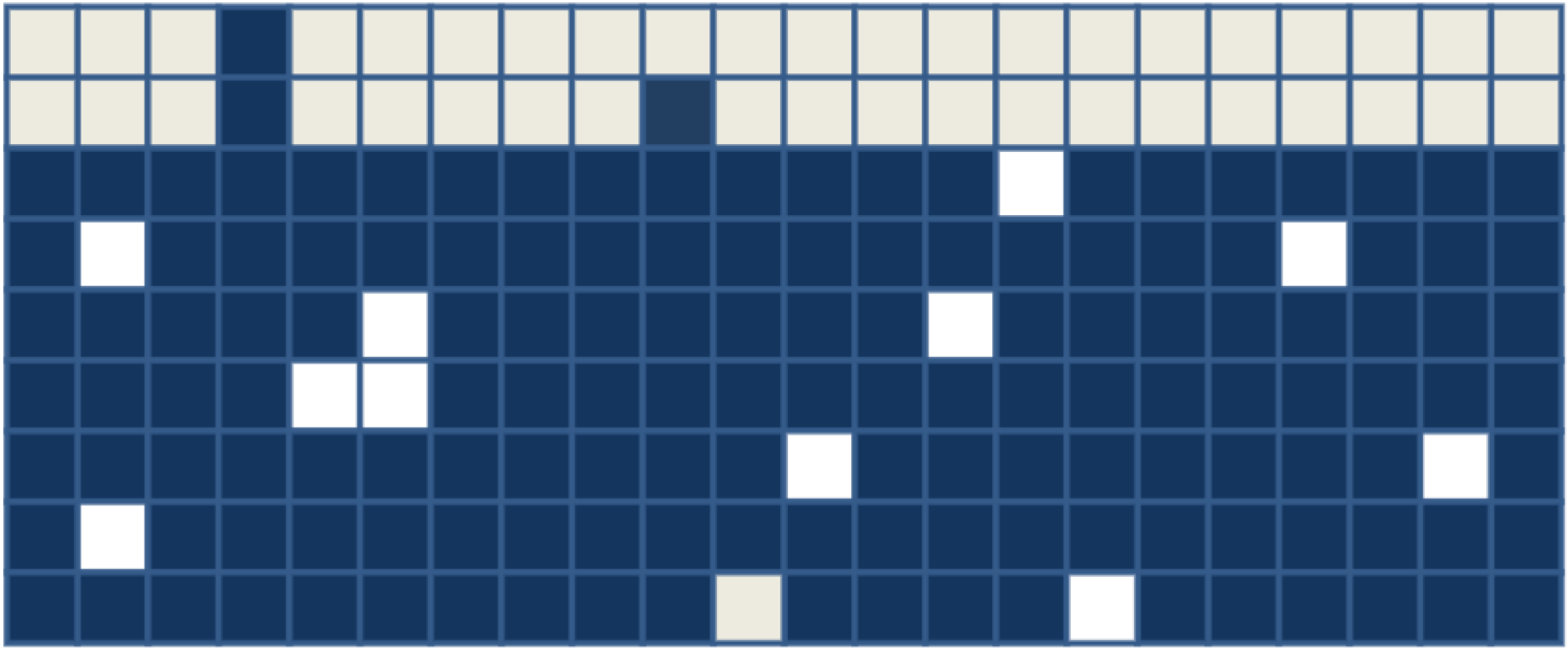
- take average user ratings into account, transform the original ratings
- U: set of users who have rated both items a and b

$$sim(a, b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

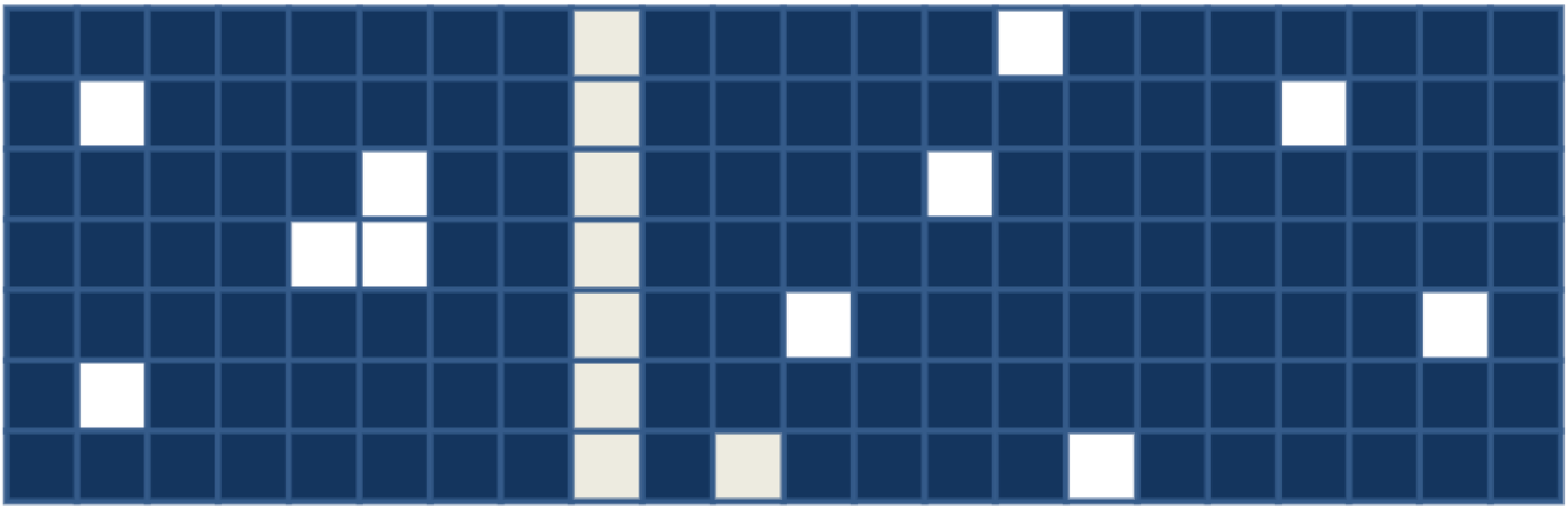
Problems with CF

- **Cold Start**: There needs to be enough other users already in the system to find a match.
- **Sparsity**: If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is hard to find users that have rated the same items.
- **First Rater**: Cannot recommend an item that has not been previously rated.
  - New items
  - Esoteric items
- **Popularity Bias**: Cannot recommend items to someone with unique tastes.
  - Tends to recommend popular items.

User cold-start



Item Cold-Start



Solutions for Cold-Start

- Use better algorithms
  - Beyond nearest-neighbour approach
  - Use weaker notions of similarity (e.g., recursive collaborative filtering)
- Matrix factorization (e.g., singular value decomposition)
- Association rule mining
- Probabilistic models
  - Clustering models, Bayesian networks,
- Various other machine learning approaches
  - Deep learning

Matrix Factorization

- Exploring latent features (e.g., attributes) of rating matrix *R* of *U* users and *D* items.
  - *K* latent features
  - P (of size U×K)
  - Q (of size D×K)

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

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj}$$

Koren et al. Matrix factorization techniques for recommender systems. Computer, 2009.
https://towardsdatascience.com/paper-summary-matrix-factorization-techniques-for-recommender-systems-82d1a7ace74

Matrix Factorization

- Exploring latent features (e.g., attributes)

Item D

User U

	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

=

U, K

A	*1.2*	*0.8*
B	*1.4*	*0.9*
C	*1.5*	*1.0*
D	*1.2*	*0.8*

X

D, K

W	X	Y	Z
1.5	*1.2*	*1.0*	*0.8*
1.7	*0.6*	*1.1*	*0.4*

Rating Matrix **R:** U×D

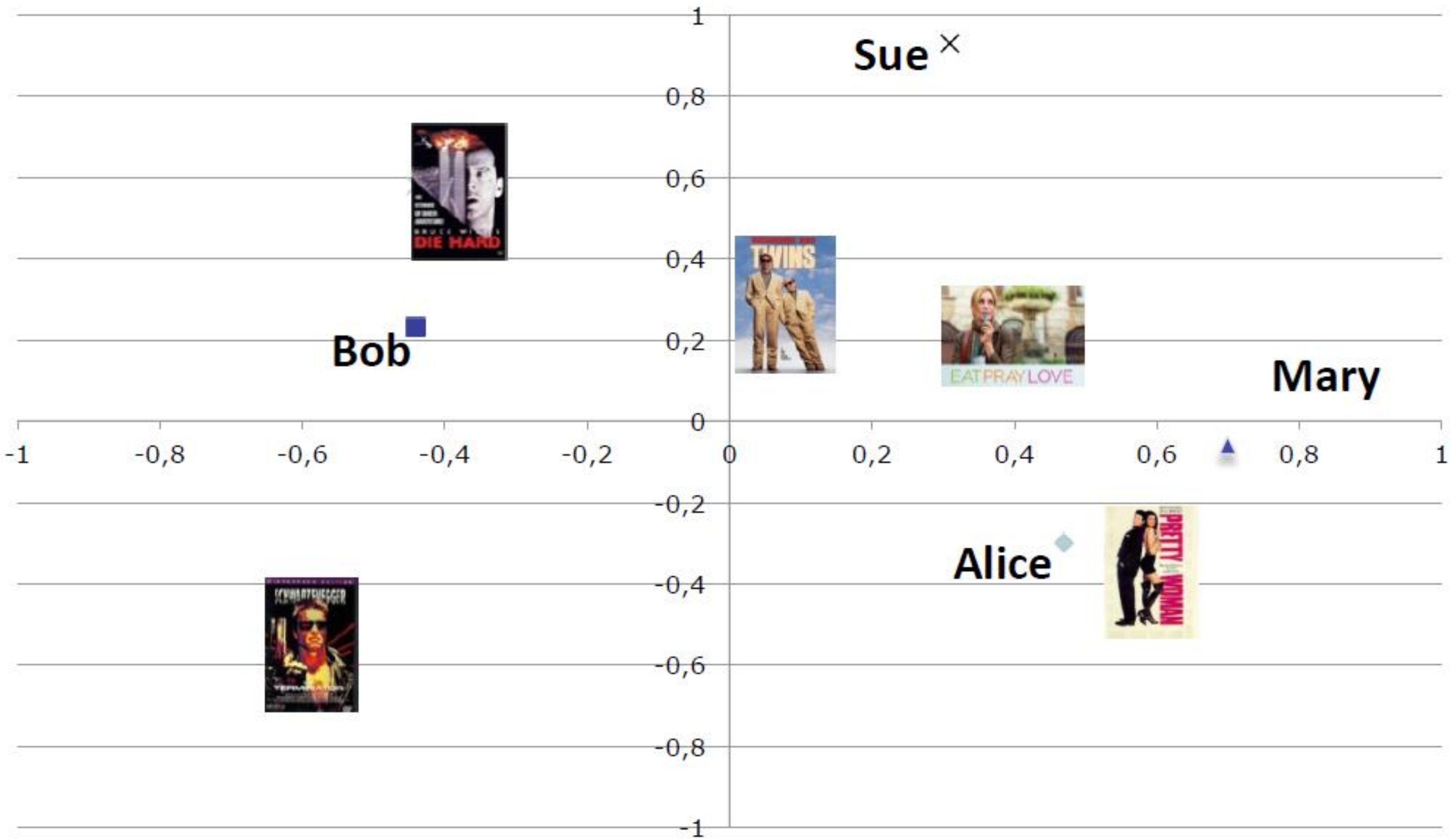
User Matrix **P:** U×K

Item Matrix **Q^T:** K×D

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj}$$

Koren et al. Matrix factorization techniques for recommender systems. Computer, 2009.
https://towardsdatascience.com/paper-summary-matrix-factorization-techniques-for-recommender-systems-82d1a7ace74

A picture says …



Matrix Factorization

- Iterative optimization through gradient descent
  - Error

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

  - Partial derivative

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2 e_{ij} q_{kj}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2 e_{ij} p_{ik}$$

  - Update rules

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

α is a constant whose value determines the **rate of approaching the minimum**.

Koren et al. Matrix factorization techniques for recommender systems. Computer, 2009.
https://towardsdatascience.com/paper-summary-matrix-factorization-techniques-for-recommender-systems-82d1a7ace74

Matrix Factorization

- Convergence

$$E = \sum_{(u_i,d_j,r_{ij})\in T_r} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T_r} (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

  - *Tr* is the training set (i.e., where r_{ij} is available).
- Regularization
  - Avoid overfitting by penalizing the magnititudes of *p* and *q*

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

Matrix Factorization

- A sample result

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

```
R = np.array([
     [5, 3, 0, 1],
     [4, 0, 0, 1],
     [1, 1, 0, 5],
     [1, 0, 0, 4],
     [0, 1, 5, 4],
])

mf = MF(R, K=2, alpha=0.1, beta=0.01, iterations=20)
```

```
[[ 4.99   3.     3.34   1.01]
 [ 4.     3.18   2.98   1.01]
 [ 1.02   0.96   5.54   4.97]
 [ 1.     0.6    4.78   3.97]
 [ 1.53   1.05   4.94   4.03]]
```

http://www.albertauyeung.com/post/python-matrix-factorization/

Collaborative Filtering - Summary

- Intuitive, well understood
- Performs well in practice
- No need for feature engineering

- Requires user community with a critical mass
- Computational challenges because of the huge matrix…
- Incorporating external information is difficult

Content-based (CB) RecSys

- Attributes/content of an item
  - Movie: actors, director, category, description, …

- Independent of the opinions of other users

- Ideas
  - Identifying items similar to those a user has rated
  - Matching an item with a user's profile

Content representation and item similarities (e.g., movies)

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, Murder, Neo-nazism
...					

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follet, ..	Paperback	25.65	detective, murder, New York

- **Simple approach**
  - Compute the similarity of an unseen item with other items in the user profile based on the keyword overlap (e.g. using Jaccard)
  - $$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Content-based RecSys - Summary

- Independent from other users (no need for critical mass)
- Recommendation can be given for a single user
- The cold start problem is smaller
- No need for handling a huge matrix
- It recommends from the long tail
- It can give you a "user model"

- Keywords/description may not be sufficient

Content-based RecSys - Summary

- Feature engineering is domain-specific and requires external data collection
- The filter bubble problem:
  - The greatest predicted rating might be a wrong recommendation as it “overfits” to the user’s preferences
    - if the user rated only Hungarian and Chinese restaurants the system won’t recommend a Greek restaurant (even it’s the best in the town)
- A new user has to be modeled, i.e. a sufficient personal training data is needed

Knowledge-based (KB) RecSys

- Products with low number of available ratings

- Time span plays an important role
  - Five-year-old ratings for computers
  - User lifestyle or family situation changes

- Customers want to define their requirements explicitly
  - The color of the car should be black.

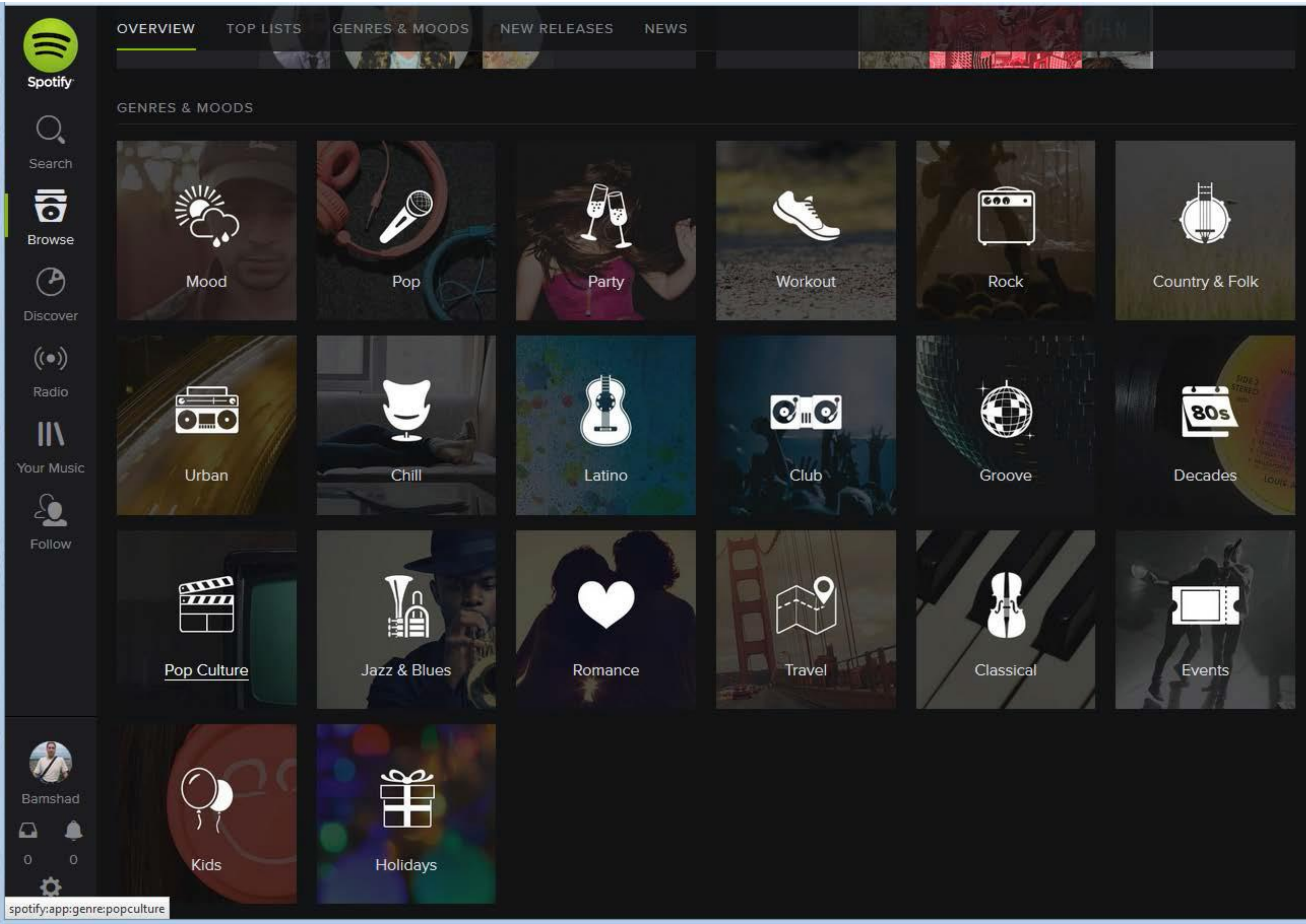
Three RecSys Paradigms

	Pros	Cons
Collaborative	No knowledge-engineering effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
Content-based	No community required, comparison between items possible	Content descriptions necessary, cold start for new users, no surprises
Knowledge-based	Deterministic recommendations, assured quality, no cold-start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

Context-aware RecSys (CARS)

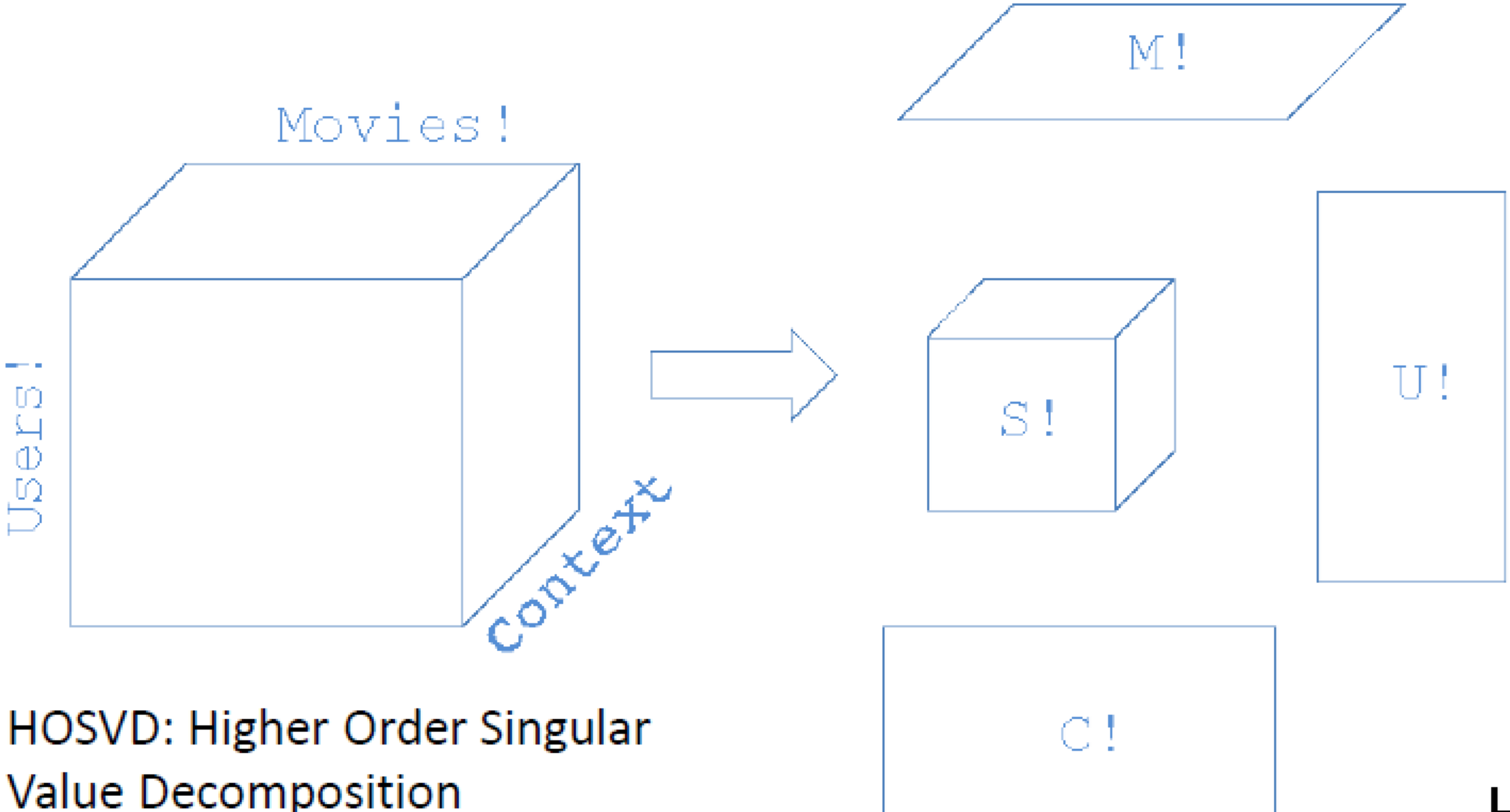
- Recommendation can also be influenced by context
  - Time, location, weather, social information, mood, device, …
  - Fine grained recommendation

- Contextual information will add extra dimensions to existing frameworks
  - Extending matrix to tensor


Spotify
OVERVIEW
TOP LISTS
GENRES & MOODS
NEW RELEASES
NEWS
GENRES & MOODS
Search
Browse
Discover
Radio
Your Music
Follow
Bamshad
0
0
Mood
Pop
Party
Workout
Rock
Country & Folk
Urban
Chill
Latino
Club
Groove
Decades
80s
Pop Culture
Jazz & Blues
Romance
Travel
Classical
Events
Kids
Holidays
spotify:app:genre:popculture

Sample Data

User	Movie	Time	Location	Companion	Rating
U1	*Titanic*	Weekend	Home	Family	4
U2	*Titanic*	Weekday	Home	Family	5
U3	*Titanic*	Weekday	Cinema	Friend	4
U1	*Titanic*	Weekday	Home	Friend	?



HOSVD: Higher Order Singular Value Decomposition

HOSVD Model

$U \in \mathbb{R}^{n \times d_U}$, $M \in \mathbb{R}^{m \times d_M}$ and $C \in \mathbb{R}^{c \times d_C}$

$S \in \mathbb{R}^{d_U \times d_M \times d_C}$

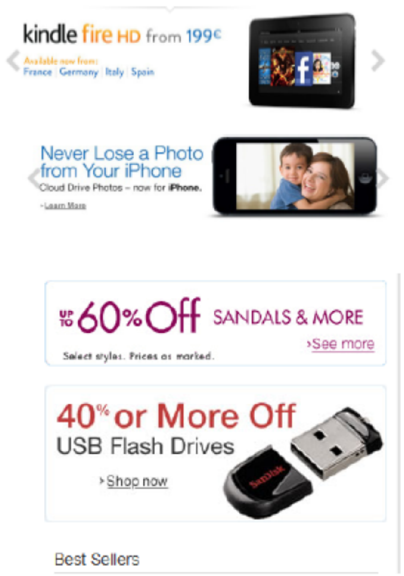
$$R[U, M, C, S] := L(F, Y) + \Omega[U, M, C] + \Omega[S]$$

Evaluation of RecSys

- What is a good recommendation?

What are the measures in practice?

- **Total sales numbers**
- **Promotion of certain items**
- **...**
- **Click-through-rates**
- **Interactivity on platform**
- **...**
- **Customer return rates**
- **Customer satisfaction and loyalty**



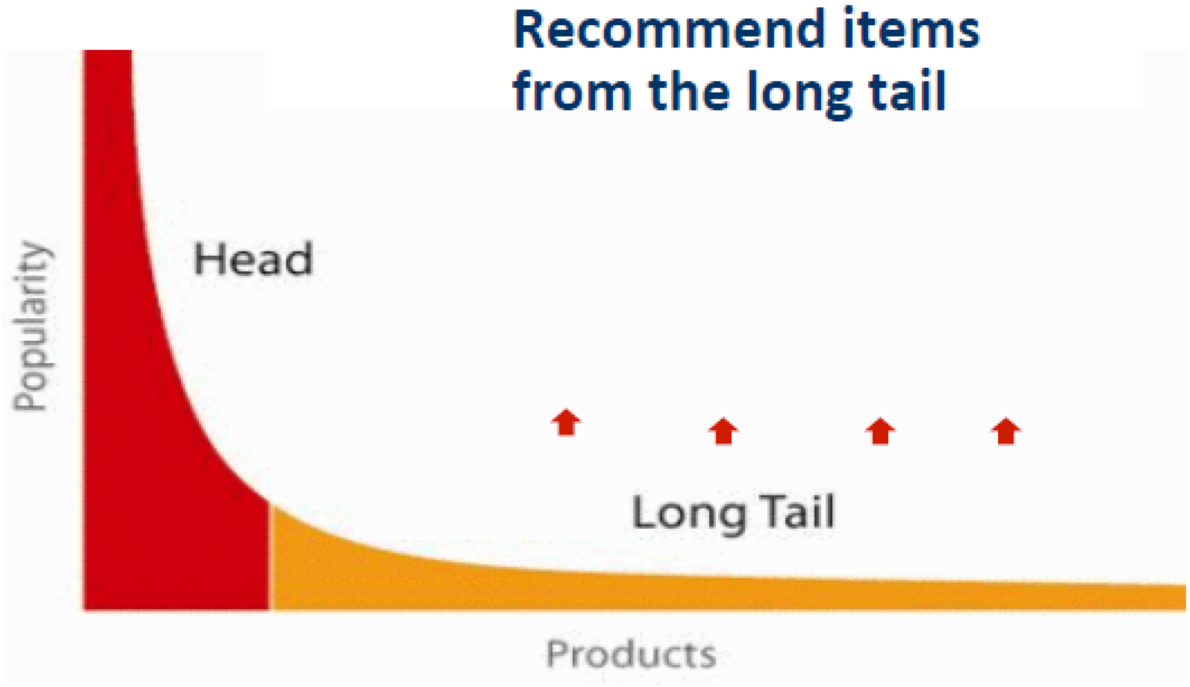
However, these evaluation methods only work for "operative" systems, where we already have active users!
What if the domain is brand-new ? (will see later)

Purpose and success criteria

- Different perspectives/aspects
  - Depends on domain and purpose
  - No holistic evaluation scenario exists

- Retrieval perspective
  - Reduce search costs
  - Provide "correct" proposals
  - Assumption: Users know in advance what they want
- Recommendation perspective
  - Serendipity – identify items from the Long Tail – not obvious recommendations!
  - Users did not know about their existence

Sample case



- "Recommend widely unknown items that users might actually like!"
- 20% of items accumulate 74% of all positive ratings

Purpose and success criteria

- Prediction perspective
  - Predict to what degree users like an item
  - Most popular evaluation scenario in research
- Interaction perspective
  - Give users a "good feeling"
  - Educate users about the product domain
  - Convince/persuade users - explain
- Finally, conversion perspective
  - Commercial situations
  - Increase "hit", "clickthrough", "lookers to bookers" rates
  - Optimize sales margins and profit

How do we know?

- **Test with real users**
  - A/B tests
  - Example measures: sales increase, click through rates – as we said, real users are often not available for new types of recommenders (e.g., recommending places to visit during a trip)
- **Laboratory studies**
  - Controlled experiments: recruit a number of possible users
  - Example measures: satisfaction with the system (questionnaires)
- **Offline experiments**
  - Based on historical data (predict the “known” future: remove items from a user’s purchase list, learn a recommendation model based on these “purged” data, and then test if system would recommend removed items)
  - Example measures: prediction accuracy, coverage

Offline experimentation needs large datasets

- Netflix prize dataset
  - Web-based movie rental
  - Prize of $1,000,000 for accuracy improvement (RMSE) of 10% compared to own Cinematch system.
- Movilens
  - 11 million ratings of 8500 movies
  - https://en.wikipedia.org/wiki/MovieLens
- Million song dataset
  - https://labrosa.ee.columbia.edu/millionsong/
- Wiki-MED
  - the largest multi-domain
  - http://iswc2018.semanticweb.org/sessions/wiki-mid-a-very-large-multi-domain-interests-dataset-of-twitter-users-with-mappings-to-wikipedia/

Need to Know

- Recommendation algorithms
  - Collaborative filtering
  - Content-based
  - Learning based
- Context Aware Recommendation
  - Time, location, …
- Evaluation

References

- D. Jannah, M. Zanker, and G. Friedrich, Recommender Systems, IJCAI 2017 Tutorial.
- B. Mobasher, Context aware Recommendation, KDD 2014 Tutorial.
- Must-read papers on Recommender System
  - https://github.com/hongleizhang/RSPapers

- [Hariri et al., 2012] Context-aware music recommendation based on latent topic sequential patterns, RecSys.

More advanced techniques

- Hashing
  - Locality sensitive hashing
  - Multi-modal hashing
- Cross-modal retrieval