Recommender Systems

Data Science Dojo

datasciencedojo
unleash the data scientist in you

Overview

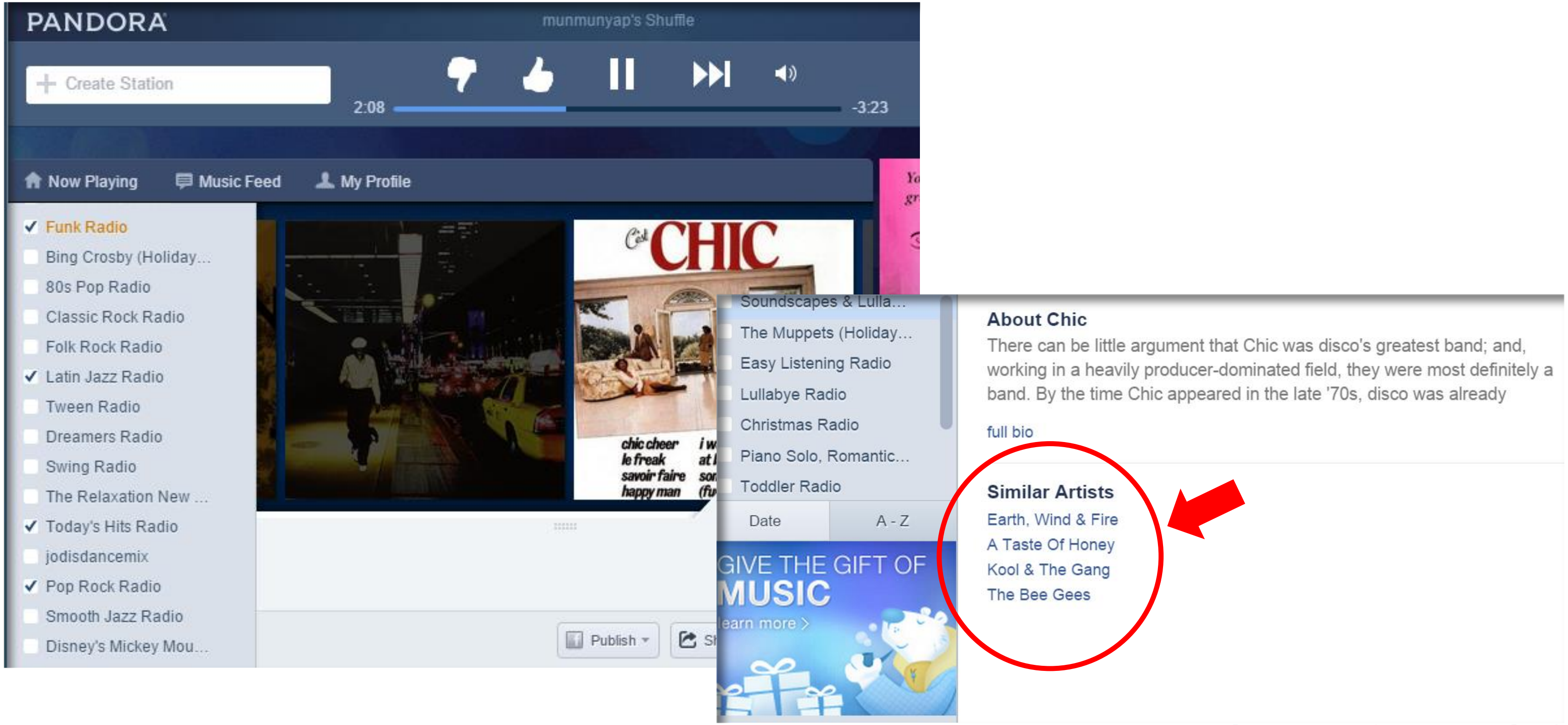
- What are Recommender Systems?
- How do they work?
  - Collaborative Recommendation
  - Content-Based Recommendation
- Example using Azure ML

Recommender Systems

- What are Recommender Systems?
  - To solve information overload problem
  - Automated systems to filter and recommend products based on users' interest and taste.

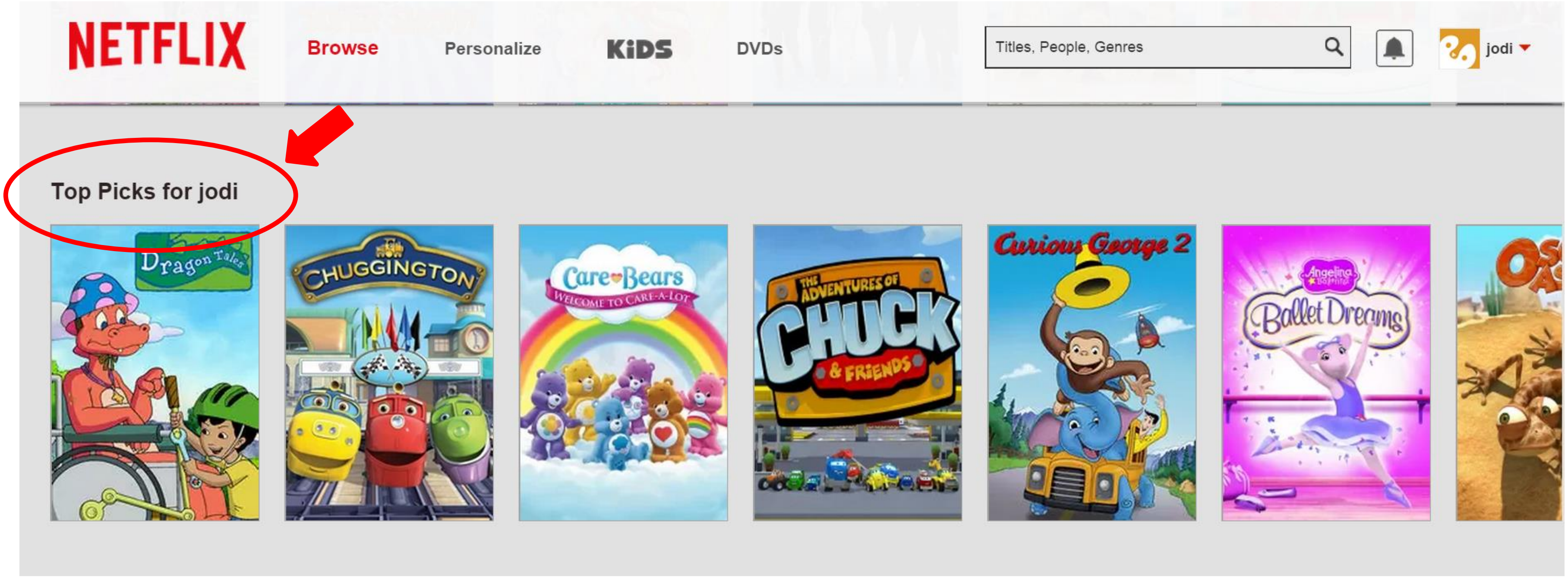
Example: Retail

Example: Entertainment



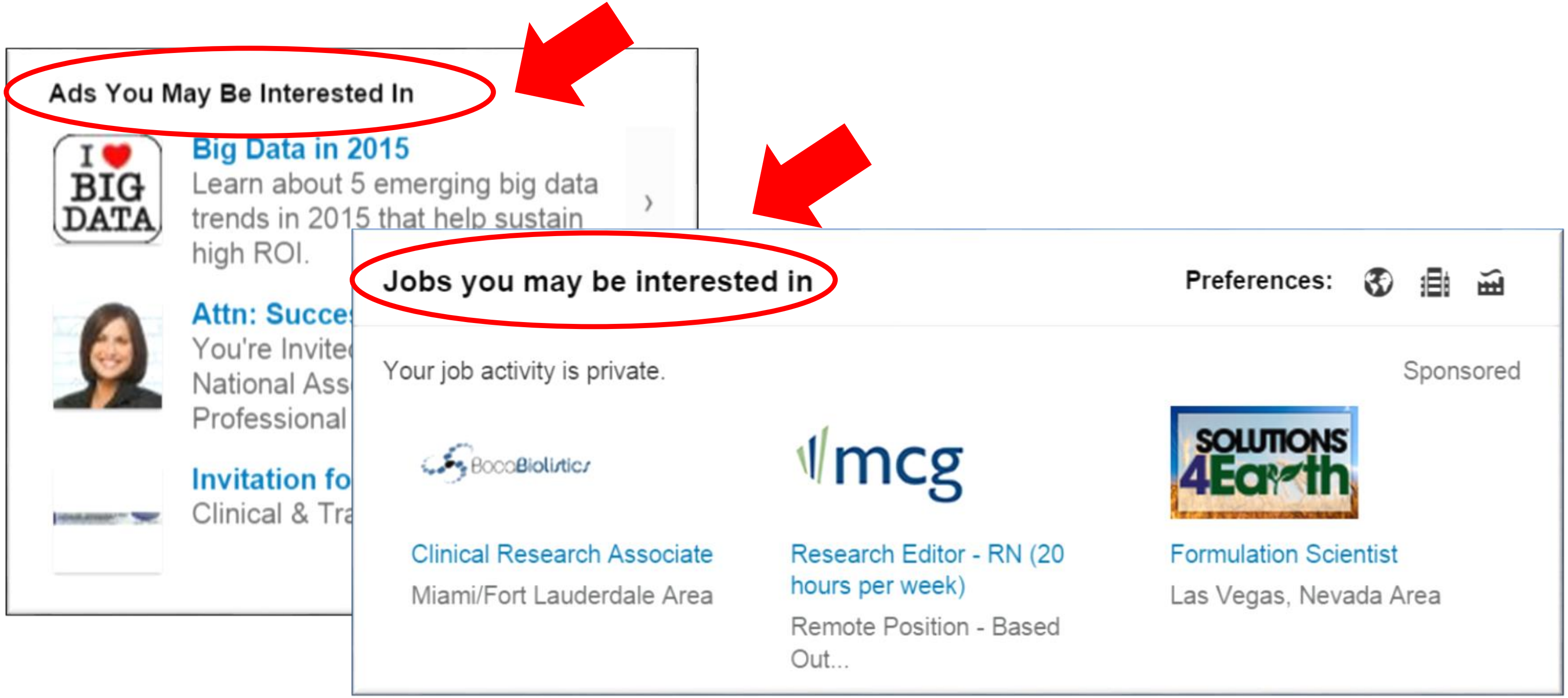
datasciencedojo
unleash the data scientist in you

Example: Entertainment



datasciencedojo
unleash the data scientist in you

Example: Social Media



Why recommendation systems?

For customer

- Narrow down the set of choices
- Discover new things
- Find things that are interesting
- Save time

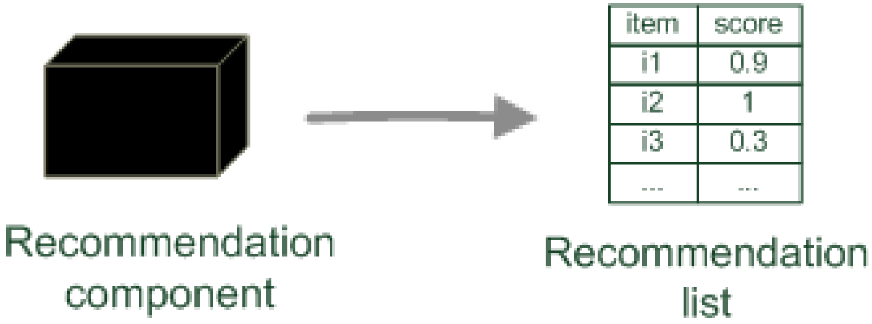
Why recommendation systems?

For businesses

- Increase the number of items sold
- Sell more diverse items
- Increase the user satisfaction
- Better understand what the user wants

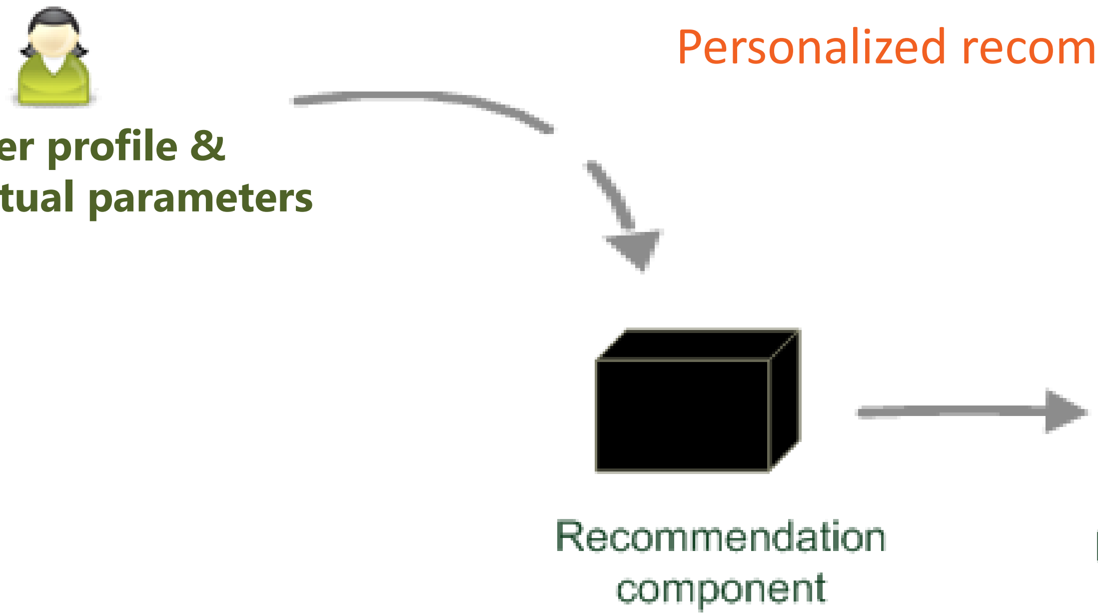
Recommender Systems

Recommender systems reduce information overload by estimating relevance



Recommender Systems

Personalized recommendations



item	score
i1	0.9
i2	1
i3	0.3
...	...

Recommendation list

Recommender Systems

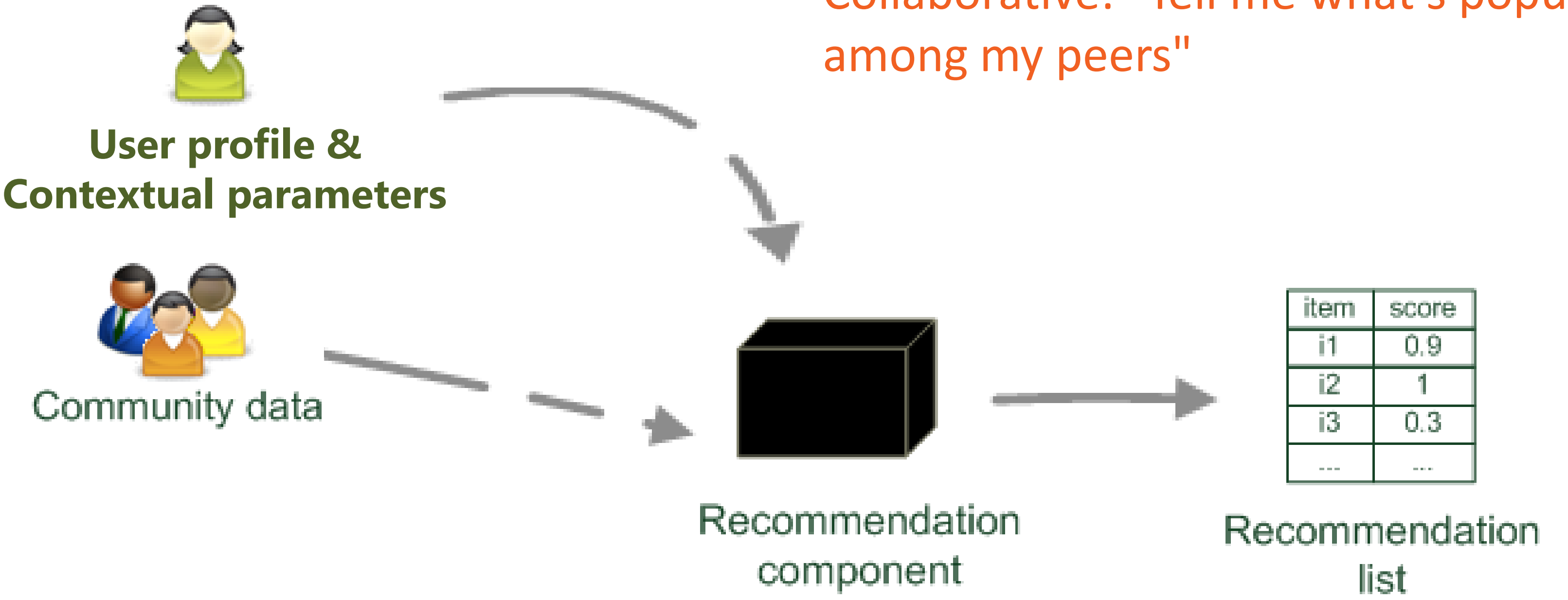
- Collaborative Recommendation
- Content-Based Recommendation

Collaborative Filtering

- Maintain a database of many users' ratings of a variety of items.
- For a given user, find other similar users whose ratings strongly correlate with the current user.
- Recommend items rated highly by these similar users, but not rated by the current user.

Collaborative Filtering (CF)

Collaborative: "Tell me what's popular among my peers"



item	score
i1	0.9
i2	1
i3	0.3
...	...

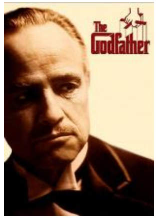
Collaborative Filtering

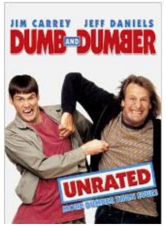
- Most popular recommendation algorithm
  - Used by large, commercial e-commerce sites
  - Well-understood, variety of algorithms
  - Applicable to many domain (books, movies, songs,…)
- Approach: borrow the “wisdom of the crowd” to recommend items

Collaborative Filtering

- Assumption:
  - Users give ratings to items
  - Users who has similar tastes in the past, have similar tastes in the future.
- User-based collaborative
- Item-based collaborative

Movie Rating Example





Alice	5	3	4	4	?
Bob	3	1	2	3	3
Chris	4	3	4	3	5
Donna	3	3	1	5	4
Evi	1	5	5	2	1

Movie Rating Example

Goal: Given Alice is an "active" user, we want to predict the rating of movie *i* Alice hasn't seen before.

- Find set of users who liked the same items as Alice in the past and also had rated movie *i*
- Predict Alice rating on movie *i*
- Repeat for all items Alice has not seen and recommend the best rated.

User-Based collaborative filtering

- How do we define similarity?
- How many neighbor should we include?
- How to generate prediction from neighbors’ ratings?

User-Based collaborative filtering

- **Nearest neighbors**
  - **Pearson correlation**

j : Alice
k: Bob
P: set of items, rated by Alice and Bob

j,k : users

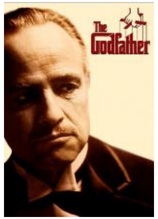
$r_{j,p}$: rating of user j for item p

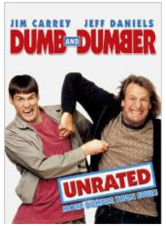
P: set of items, rated both by j and k

Possible similarity values between -1 and 1

$$\boldsymbol{sim(j,k)} = \frac{\sum_{p\,\in \boldsymbol{P}}(\boldsymbol{r}_{j,p} - \bar{\boldsymbol{r}}_j)(\boldsymbol{r}_{k,p} - \bar{\boldsymbol{r}}_k)}{\sqrt{\sum_{p\,\in \boldsymbol{P}}(\boldsymbol{r}_{j,p} - \bar{\boldsymbol{r}}_j)^{\boldsymbol{2}}}\sqrt{\sum_{\boldsymbol{p}\,\in \boldsymbol{P}}(\boldsymbol{r}_{k,p} - \bar{\boldsymbol{r}}_k)^{\boldsymbol{2}}}}$$

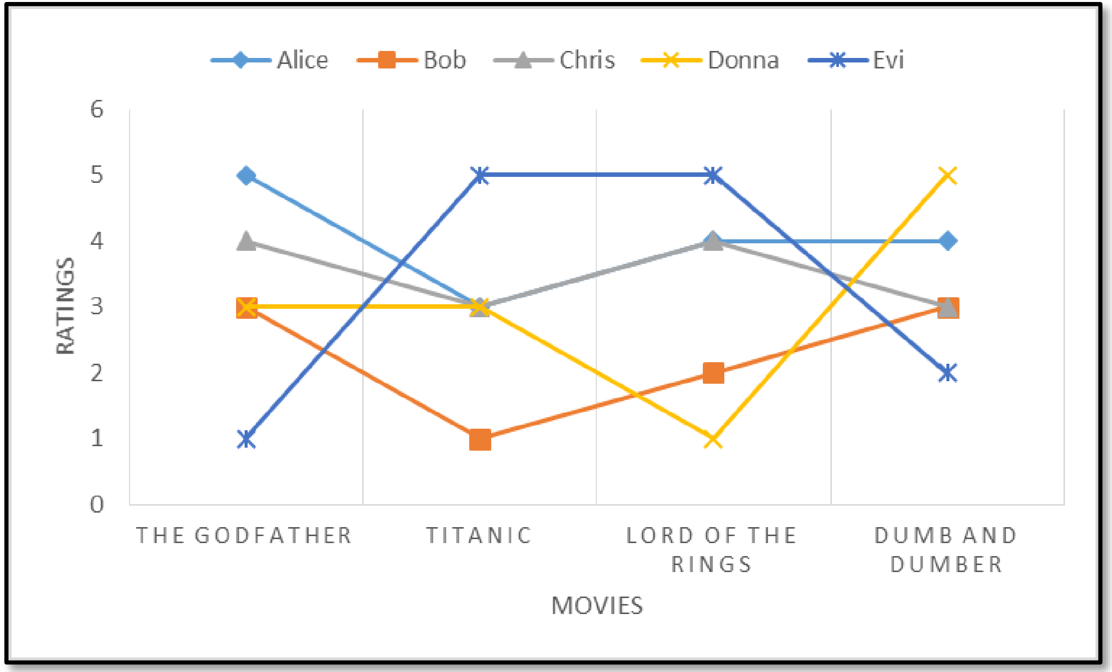
Pearson Correlation





Alice	5	3	4	4	?	
Bob	3	1	2	3	3	sim=0.85
Chris	4	3	4	3	5	sim=0.90
Donna	3	3	1	5	4	sim=0.70
Evi	1	5	5	2	1	sim=0.79

Pearson Correlation



datasciencedojo
unleash the data scientist in you

Making prediction

$$\boldsymbol{pred(j,i)} = \bar{\boldsymbol{r}}_j + \frac{\sum_{\boldsymbol{k} \in N} \boldsymbol{sim(j,k)} * (\boldsymbol{r}_{\boldsymbol{k,i}} - \overline{\boldsymbol{r}_{\boldsymbol{k}}})}{\sum_{k \in N} \boldsymbol{sim}(j,k)}$$

j : Alice
k: Bob
i: movie Spirited Away

- Calculate, whether the neighbors' ratings for the unseen item i are higher or lower than their average
- Combine the rating differences – use the similarity with j user as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

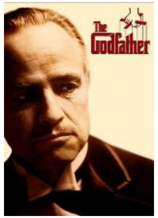
Making recommendations

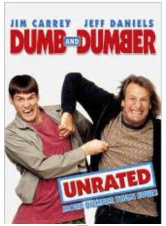
- Making predictions is typically not the ultimate goal
- Usual approach
  - Rank items based on their predicted ratings
- However
  - This might lead to the inclusion of (only) niche items
- Better approach
  - Optimize according to a given rank evaluation metric

Item-based collaborative filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for movies that are similar to movie 5
  - Take Alice's ratings for these items to predict the rating for movie 5

Movie Rating Example





Alice	5	3	4	4	?
Bob	3	1	2	3	3
Chris	4	3	4	3	5
Donna	3	3	1	5	4
Evi	1	5	5	2	1

Other similarity measurement

- cosine similarity

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

- Adjusted cosine similarity

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

datasciencedojo

unleash the data scientist in you

Collaborative Filtering Issues

- **Pros:**
  - well-understood, works well in some domains, no knowledge engineering required, serendipity of results
- **Cons:**
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results

Content-based recommendation

Goal: To learn user preferences

Recommend items that are “similar” to the user preferences

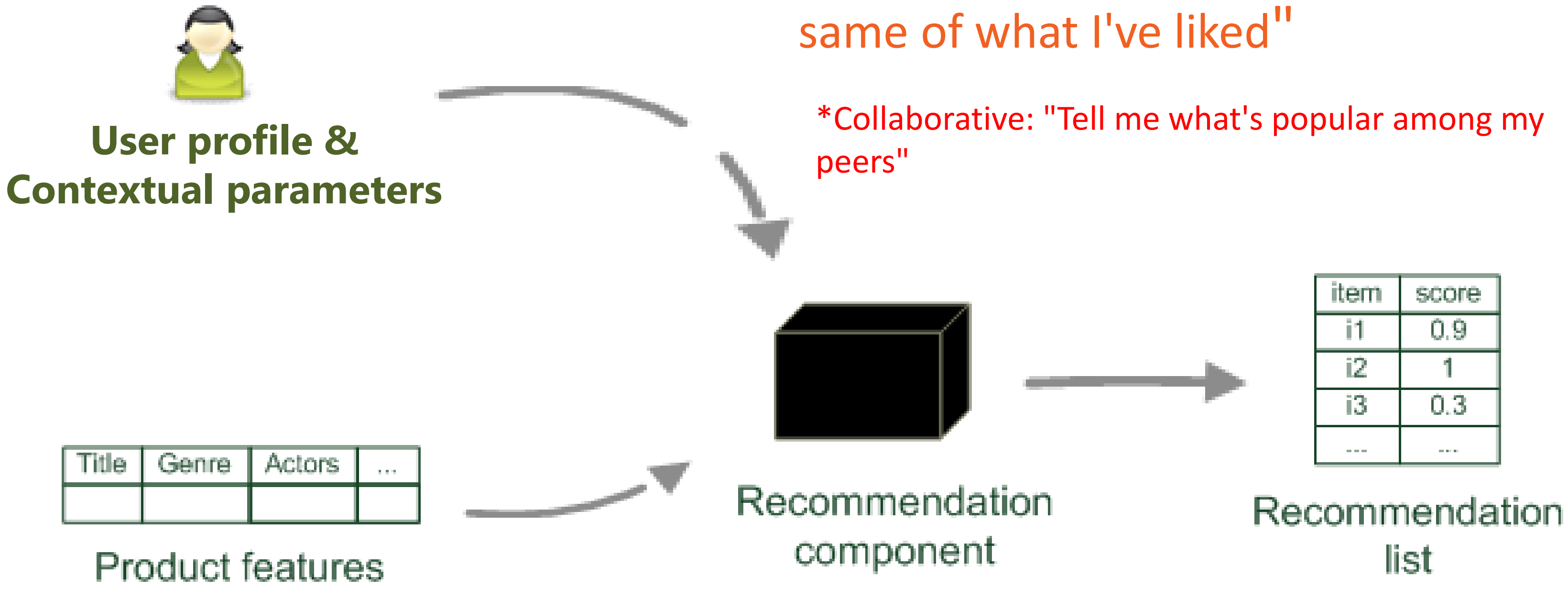
What do we need:

- Content of the items
- User profiles describing the preferences of the user.

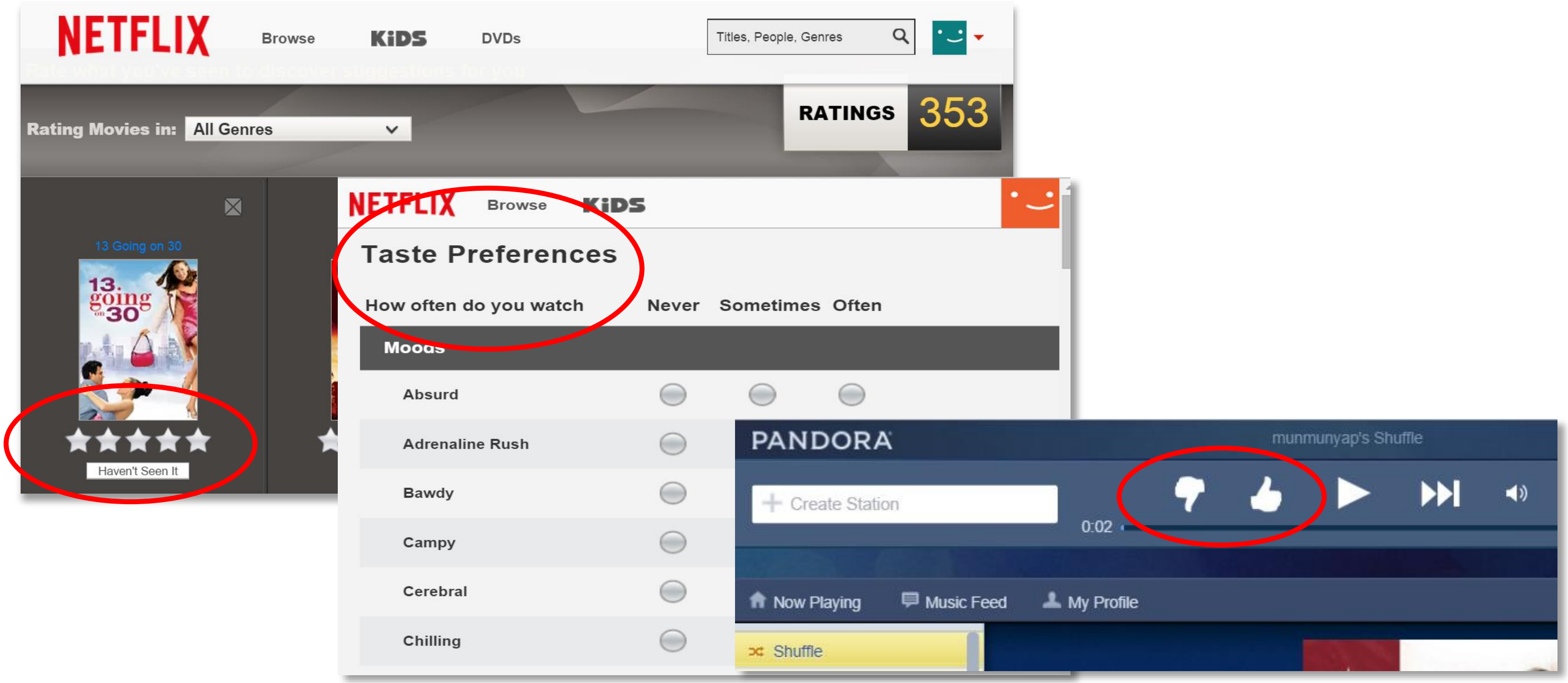
Content-based recommendation

Content-based: "Show me more of the same of what I've liked"

*Collaborative: "Tell me what's popular among my peers"



Content-based recommendation



datasciencedojo
unleash the data scientist in you

Example of “content” ?

- **Information Retrieval (IR) based method**
  - Goal is to find and rank relevant text documents (news articles, web pages)
  - Based on keywords
  - No expert recommendation knowledge involved

Content representation

Item content

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
...					

User's preferred content

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

Content representation

- **Simple approach**
  - Compute the similarity of an unseen item with the user profile based on the keyword overlap
  - $\text{sim}(b_i, b_j) = \dfrac{2 * |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$

Issues with simple keyword count

- Simple keyword representation has its problems in particular when automatically extracted because
  - Not every word has similar importance
  - Longer documents have a higher chance to have an overlap with the user profile

TF-IDF

- Standard measure: TF-IDF
  - **TF:** Measures, how often a term appears (density in a document)
    - Assuming that important terms appear more often
    - Normalization has to be done in order to take document length into account
  - **IDF:** Aims to reduce the weight of terms that appear in all documents

TF-IDF

- **Term frequency (TF)**
  - Let *freq(t,d)* number of occurrences of keyword *t* in document *d*
  - Let *max{freq(w,d)}* denote the highest number of occurrences of another keyword of *d*
  - $TF(t,d) = \frac{freq(t,d)}{\max\{freq(w,d): w \in d\}}$

IDF

- **Inverse Document Frequency (IDF)**
  - N: number of all recommendable documents
  - n(t): number of documents in which keyword t appears
  - $IDF(t) = log\frac{N}{n(t)}$

datasciencedojo
unleash the data scientist in you

TF-IDF

- Compute the overall importance of keywords
  - Given a keyword *t* and a document *d*

$$TF\text{-}IDF\,(t,d) = TF(t,d) * IDF(t)$$

Example of TF

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Example taken from http://informationretrieval.org

TF-IDF weights

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4					
Caesar	232					
Calpurnia	0					
Cleopatra	57					
mercy	1.51					
worser	1.37					

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Example taken from http://informationretrieval.org

Recommending items

- Simple method: nearest neighbors
  - Given a set of documents D already rated by the user (like/dislike, ratings)
    - Find the n nearest neighbors of a not-yet-seen item *i* in D
    - Take these ratings to predict a rating/vote for *i*

Example taken from http://informationretrieval.org

Recommending items

User's content

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95
Rating	4	3	2	5	1	3

Potential items to recommend

	The Hobbit
Bilbo	8.8
Gandalf	7.4
dwarf	4
Bombur	2.3
goblin	2.85
spider	1.51
Belladonna	0.3
Rating	?

Example taken from http://informationretrieval.org

Recommending items

- Query-based retrieval: Rocchio's method
- Probabilistic methods
- linear classification/regression algorithms
- etc

Example taken from http://informationretrieval.org

Content-based recommenders

Advantages

- No community required. Only have to analyze the items and user profile for recommendation.
- Transparency: CB method can tell you they recommend you the items based on features not other users.
- No cold start: new items can be suggested before being rated by a substantial number of users.

Content-based recommenders

Disadvantages

- Limited content analysis: required well annotated content for good recommendations.
- Over-specialization: no surprises
- New user: limited user information results in bad recommendation.

Evaluating Recommendation

- Among many techniques
  - Which one is the best in a given application domain?
  - What are the success factors of different techniques?
  - Comparative analysis based on an optimality criterion?

Evaluating Recommendation

- Research questions are:
  - Is a RS efficient with respect to a specific criteria like accuracy, user satisfaction, response time, serendipity, online conversion, ramp-up efforts, ….
  - Do customers like/buy recommended items?
  - Do customers buy items they otherwise would have not?
  - Are they satisfied with a recommendation after purchase?

Evaluating Recommendation

- Metrics measure error rate
  - **Mean Absolute Error (*MAE*)** computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | p_i - r_i |$$

  - **Root Mean Square Error (*RMSE*)** is similar to *MAE*, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_i - r_i)^2}$$

datasciencedojo
unleash the data scientist in you

Metrics

- **Discounted cumulative gain (DCG)**
  - Logarithmic reduction factor

$$DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i}$$

Where:

- *pos* denotes the position up to which relevance is accumulated
- rel_i returns the relevance of recommendation at position i

Metrics

- **Discounted cumulative gain (DCG)**
  - Logarithmic reduction factor

$$DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i}$$

Where:

- *pos* denotes the position up to which relevance is accumulated
- rel_i returns the relevance of recommendation at position i

Metrics

- **Idealized discounted cumulative gain (IDCG)**
  - Assumption that items are ordered by decreasing relevance

$$IDCG_{pos} = rel_1 + \sum_{i=2}^{|h|-1} \frac{rel_i}{\log_2 i}$$

- **Normalized discounted cumulative gain (nDCG)**

$$nDCG_{pos} \frac{DCG_{pos}}{IDCG_{pos}}$$

  - Normalized to the interval [0..1]

QUESTIONS

datasciencedojo
unleash the data scientist in you