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
- How do we evaluate them?
- Example using Azure ML

Overview

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

Recommender Systems

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

Why recommendation systems?

For customer

- Narrow down the set of choices
- Discover new, interesting things
- Save time

Why recommendation systems?

For businesses

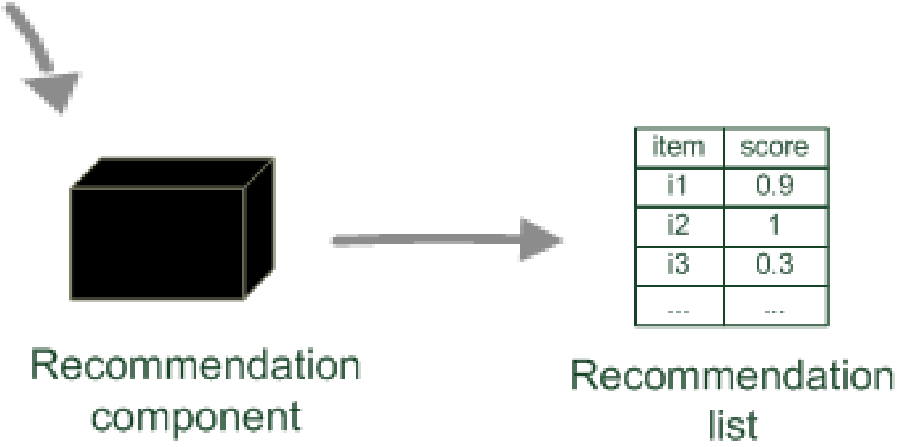
- Increase the number of items sold
- Sell more diverse items
- Better understand what the user wants
- Increase user satisfaction

Recommender Systems

User profile &
Contextual parameters

Recommender systems reduce information overload by estimating relevance

Personalized recommendations

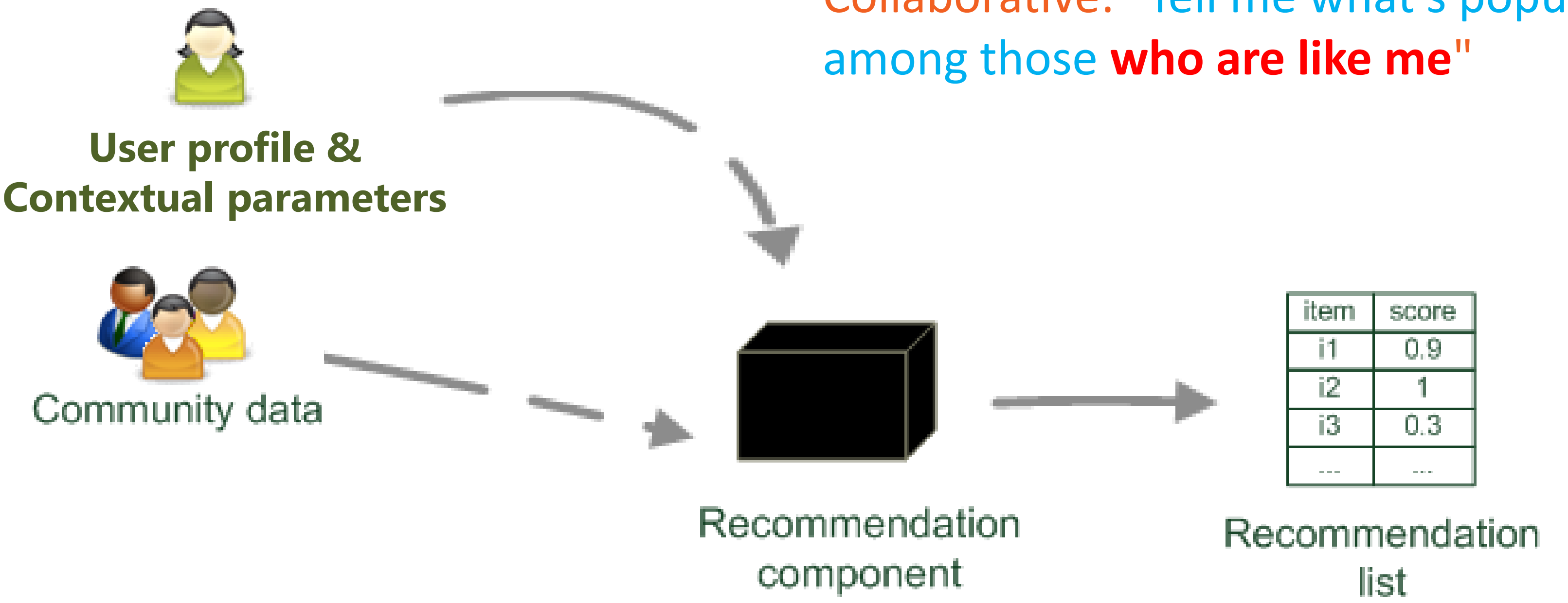


Overview

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

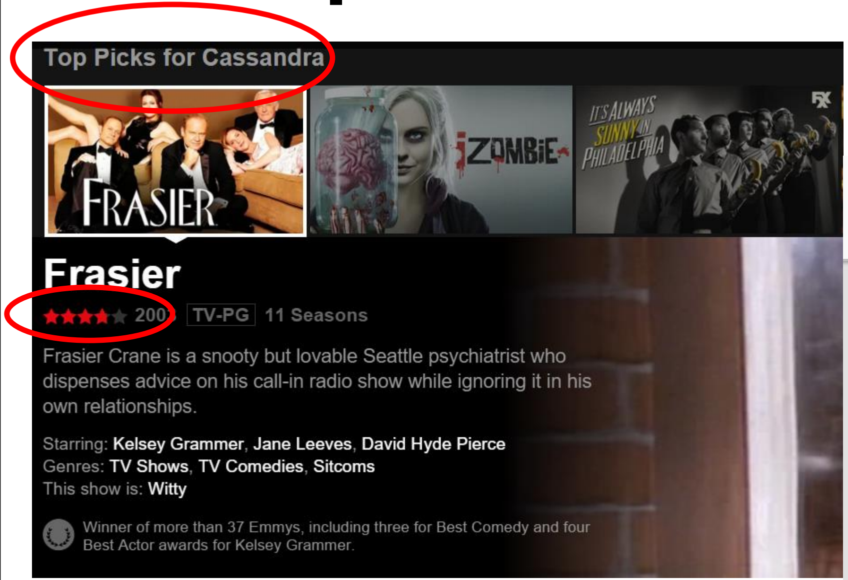
Collaborative Filtering (CF)

Collaborative: "Tell me what's popular among those **who are like me**"



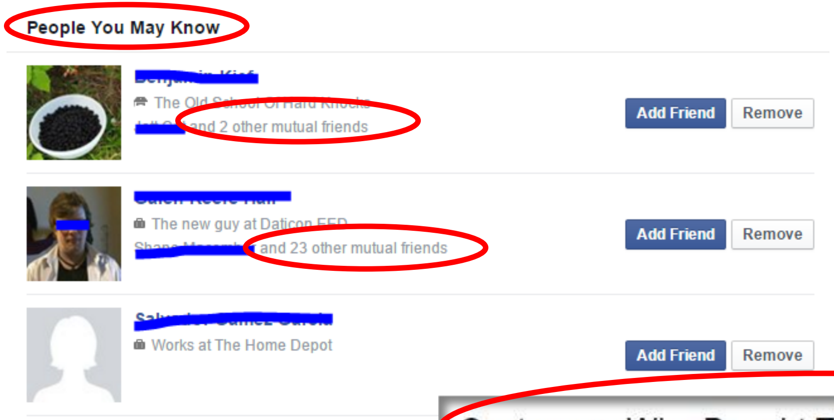
datasciencedojo
unleash the data scientist in you

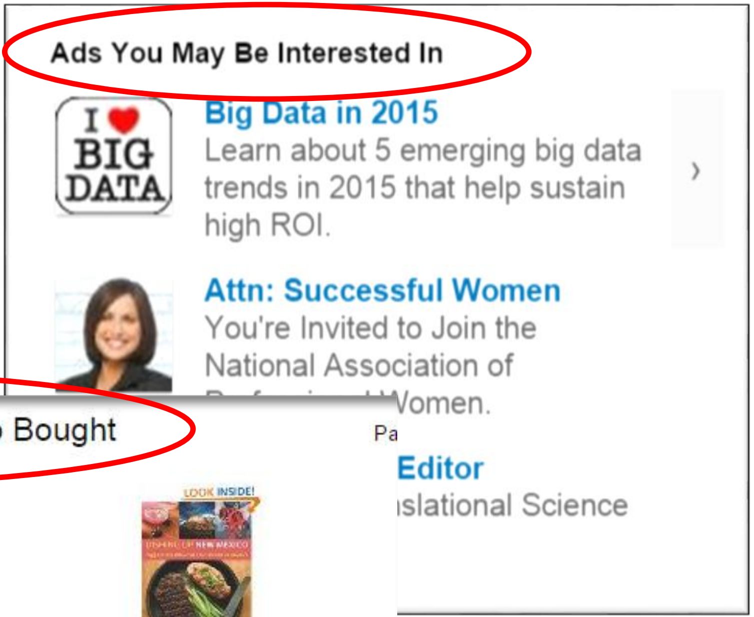
Example: Netflix



datasciencedojo
unleash the data scientist in you

Example: Retail & Social Media





datasciencedojo
unleash the data scientist in you

Collaborative Filtering

- Most popular recommendation algorithm
  - Used by large, commercial e-commerce sites
  - Well-understood, variety of algorithms
  - Applicable to many domains (books, movies, songs,…)

- Approach: borrow the "wisdom of the crowd" to recommend items

Collaborative Filtering

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

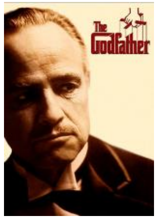
datasciencedojo
unleash the data scientist in you

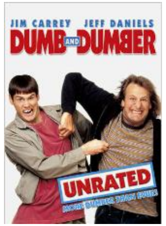
Collaborative Filtering

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

datasciencedojo
unleash the data scientist in you

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

datasciencedojo
unleash the data scientist in you

User-Based Collaborative Filtering

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

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

datasciencedojo
unleash the data scientist in you

User-Based Collaborative Filtering

- How many neighbors should we include?
  - Choose a number – depends on size of data
- How do we define similarity?
- How to do we generate predictions from the neighbors' ratings?

datasciencedojo
unleash the data scientist in you

Similarity Measurement

- ## Pearson correlation

j,k : users

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

$\bar{\boldsymbol{r}}_j$ and $\bar{\boldsymbol{r}}_k$ are the average ratings of user j and user k over all items

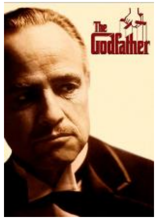
P: set of items, rated both by j and k

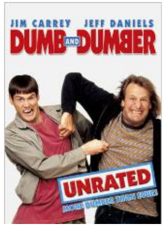
Possible similarity values between -1 and 1

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

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

Pearson Correlation

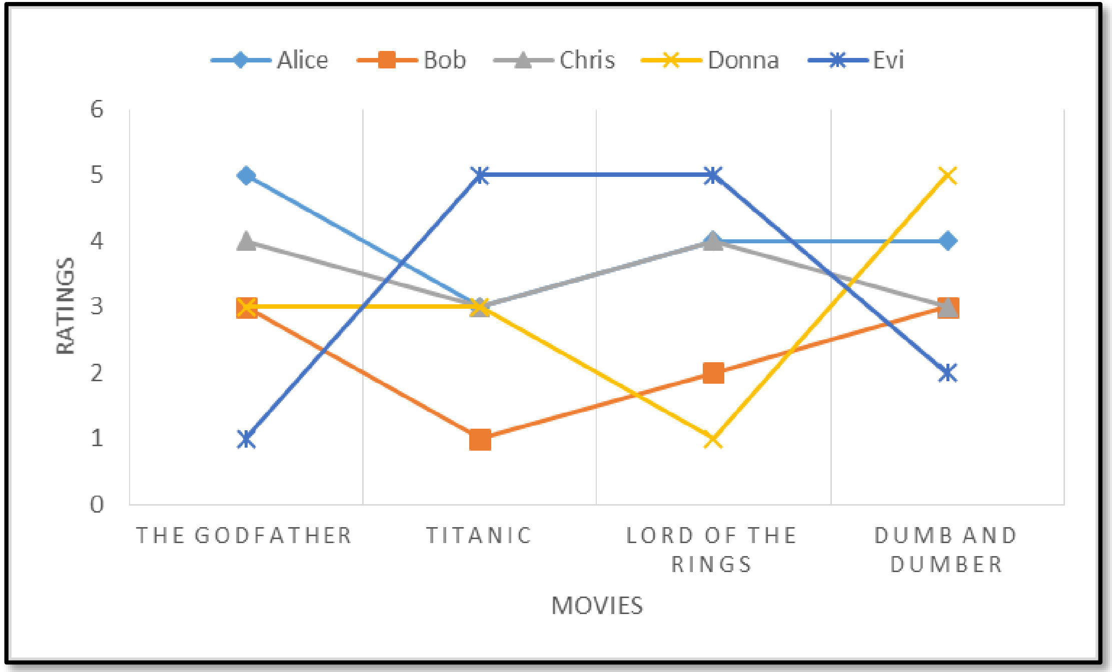




Alice	5	3	4	4	?	
Bob	3	1	2	3	3	sim=?
Chris	4	3	4	3	5	sim=?
Donna	3	3	1	5	4	sim=?
Evi	1	5	5	2	1	sim=?

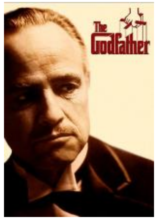
datasciencedojo
unleash the data scientist in you

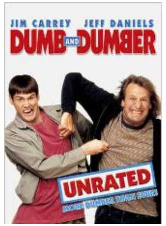
Pearson Correlation



datasciencedojo
unleash the data scientist in you

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

datasciencedojo
unleash the data scientist in you

Making Predictions

- Use "Aggregation Function"
  - Choose N neighbors
  - Simple
    - $r_{j,p} = \frac{1}{N}\sum_{k\in U} r_{k,p}$
  - Weighted & Centered
    - $r_{j,p} = \bar{\boldsymbol{r}}_j + \alpha \sum_{k\in U} simil(j,k)(r_{k,p} - \overline{\boldsymbol{r}_k})$

datasciencedojo
unleash the data scientist in you

Making recommendations

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

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**

datasciencedojo
unleash the data scientist in you

Item-based collaborative filtering

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

Similarity Measurement

- Cosine similarity

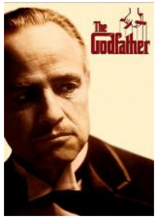
$$\boldsymbol{sim(\vec{a},\vec{b}) = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}| * |\vec{b}|} = \frac{\sum_{u\in U} r_{u,a} * r_{u,b}}{\sqrt{\sum_{u\in U} {r_{u,a}}^2}\sqrt{\sum_{u\in U} {r_{u,b}}^2}}}$$

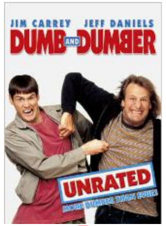
- Adjusted cosine similarity

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

datasciencedojo
unleash the data scientist in you

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

sim=0.99 sim=0.74 sim=0.72 sim=0.93

datasciencedojo
unleash the data scientist in you

Making Predictions

- Sum over items rather than users
  - Simple
    - $r_{j,p} = \frac{1}{N} \sum_{q \in P} r_{j,q}$
  - Weighted & Centered
    - $r_{j,p} = \overline{\boldsymbol{r}_p} + \alpha \sum_{q \in P} simil(p, q)(r_{j,q} - \overline{\boldsymbol{r}_q})$

datasciencedojo
unleash the data scientist in you

Collaborative Filtering Pros

- **Wide applicability**
  - Usable in wildly different domains
- **Well-understood**
  - Most well studied type of recommender
- **Simple**
  - No knowledge engineering required
- **Serendipity**
  - Odd recommendations that are very good

Collaborative Filtering Cons

- **Data sparsity & Cold Start**
  - New users need to indicate preferences for sufficient number of items before recommendations are good
  - Need initial customer/rating database
- **Scalability**
  - Millions of customers (M) and millions of items (N)
- **Grey Sheep and Black Sheep**
  - Grey sheep are users with inconsistent recommendations.
  - Black sheep are the users with idiosyncratic preferences.

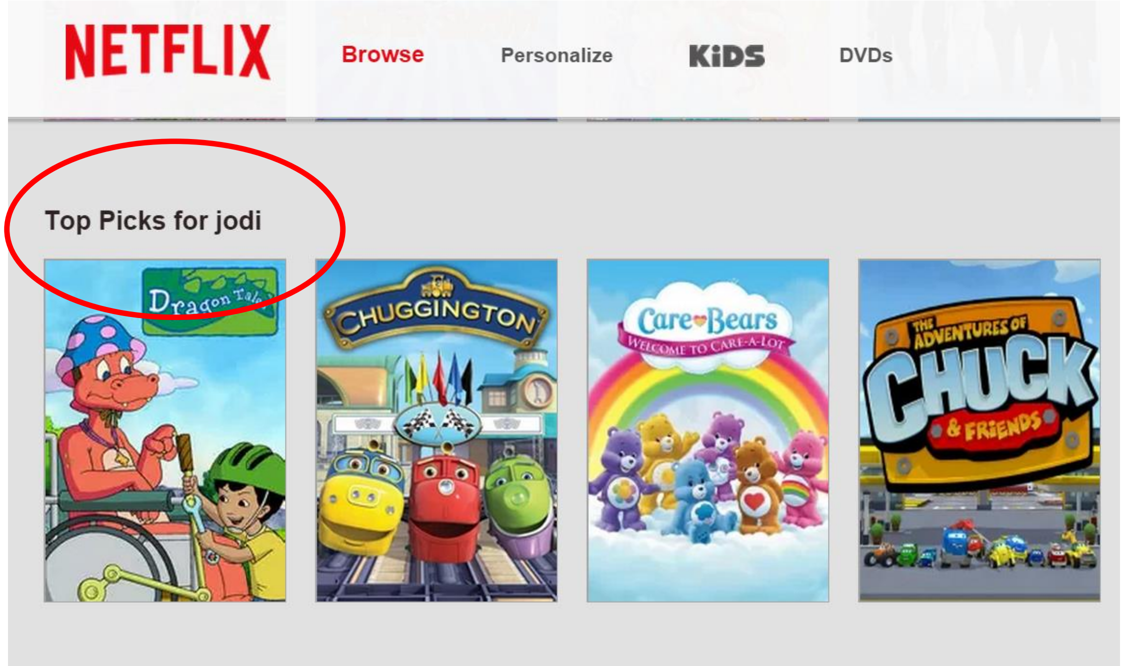
datasciencedojo
unleash the data scientist in you

Collaborative Filtering Cons

- **Shilling**
  - Intentional manipulation of ratings of your own products and competitors products
- **Diversity and Long Tail**
  - Rich tend to get richer

datasciencedojo
unleash the data scientist in you

Back to Netflix



datasciencedojo
unleash the data scientist in you

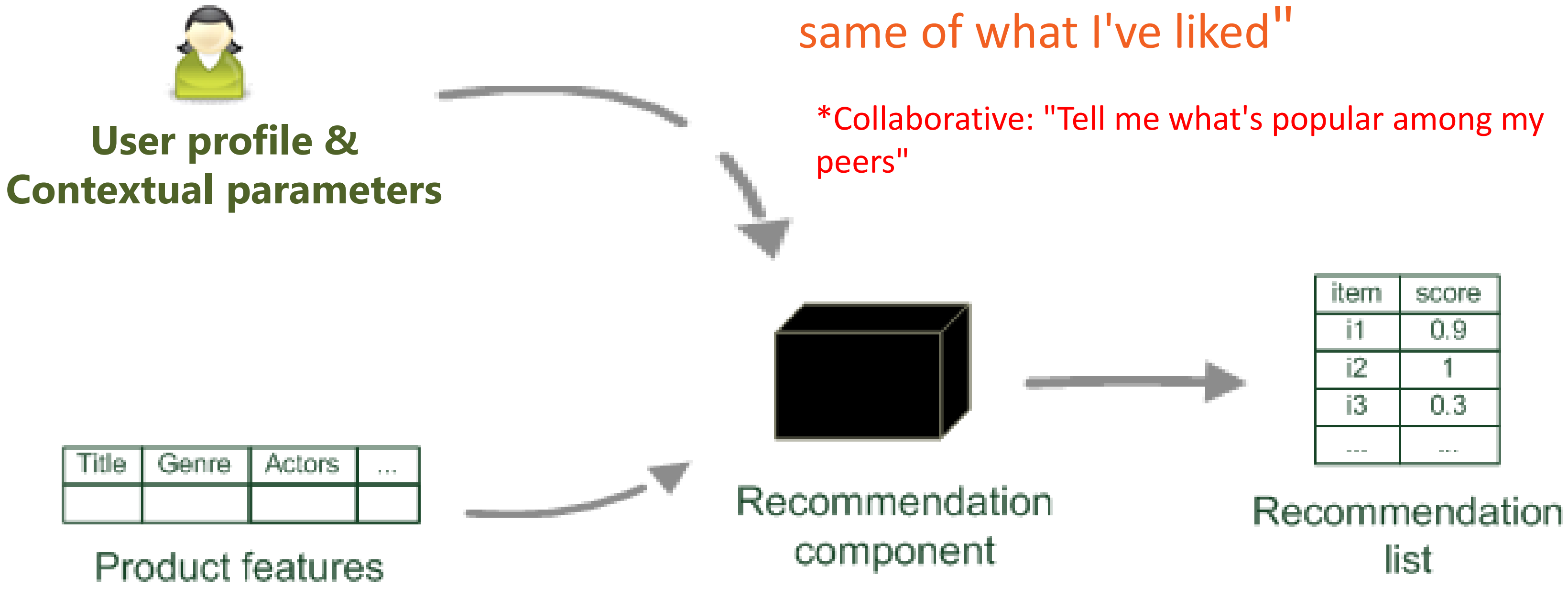
Overview

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

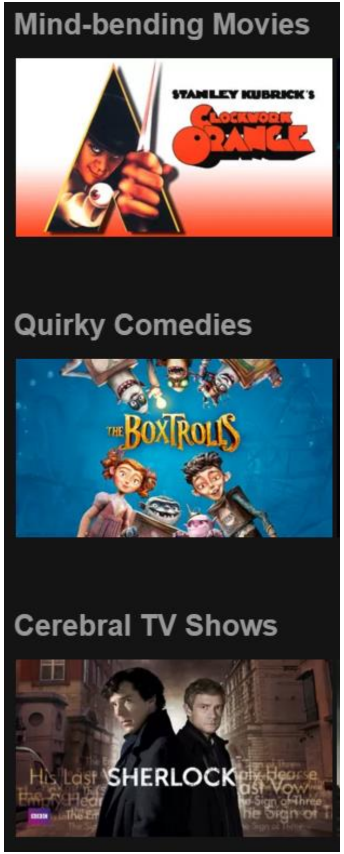
Content-based recommendation

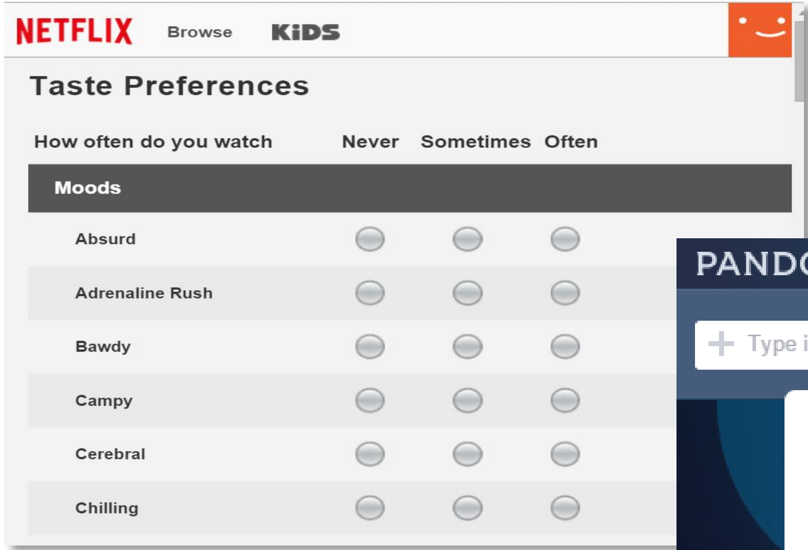
Content-based: "Show me more of the same of what I've liked"

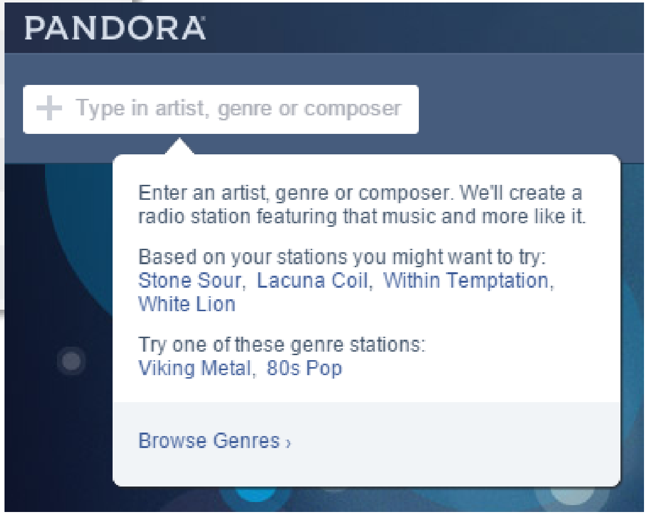
*Collaborative: "Tell me what's popular among my peers"



Examples

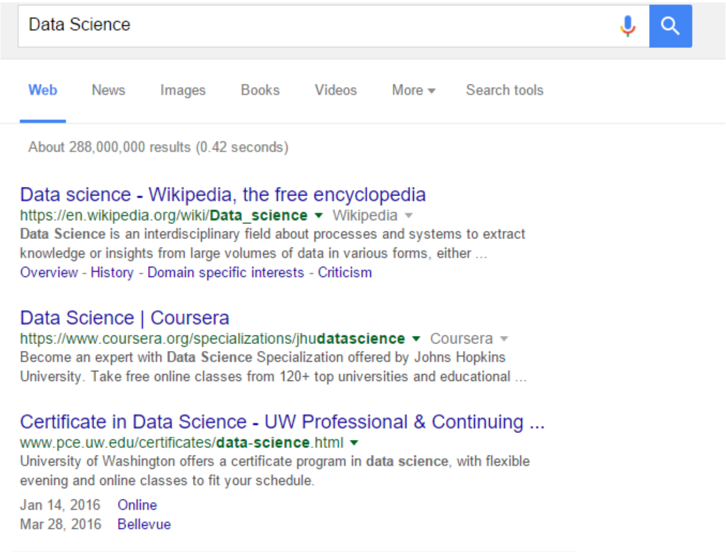






datasciencedojo
unleash the data scientist in you

Examples



datasciencedojo
unleash the data scientist in you

Content-based recommendation

Recommend items that are “similar” to the user preferences

What do we need?

- Item Profiles: list of content-based keywords
- User profiles: preferences of the user.
  - User specified or based on past behavior

Item Profile Strategies

- **Expert Labeling**
  - Assign keywords based on content
  - Good for songs, movies, etc
  - May be provided by creators/distributors
  - Crowd sourcing?

Item Profile Strategies

- **Automated Indexing**
  - Used for text documents (web pages, books, tweets)
  - Based on word content of document set
  - No expert knowledge involved
  - Can be keyword or full dictionary based

datasciencedojo
unleash the data scientist in you

Content-based recommendation

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

Simple approach: drawbacks

- Not every word has similar importance
- Longer documents have a higher chance to have an overlap with the user profile
- **Solution:** TF-IDF

datasciencedojo
unleash the data scientist in you

TF-IDF Search Exercise

- **D1** = "If it walks like a duck and quacks like a duck, it must be a duck."
- **D2** = "Beijing Duck is mostly prized for the thin, crispy duck skin with authentic versions of the dish serving mostly the skin."
- **D3** = "Bugs' ascension to stardom also prompted the Warner animators to recast Daffy Duck as the rabbit's rival, intensely jealous and determined to steal back the spotlight while Bugs remained indifferent to the duck's jealousy, or used it to his advantage. This turned out to be the recipe for the success of the duo."
- **D4** = "6:25 PM 1/7/2007 blog entry: I found this great recipe for Rabbit Braised in Wine on cookingforengineers.com."
- **D5** = "Last week Li has shown you how to make the Sechuan duck. Today we'll be making Chinese dumplings (Jiaozi), a popular dish that I had a chance to try last summer in Beijing. There are many recipies for Jiaozi."

- **Dictionary:** {beijing, dish, duck, rabbit, recipe}

Query: "Beijing duck recipe"

datasciencedojo
unleash the data scientist in you

Document Matrix

Query: "Beijing duck recipe"

	Beijing	Dish	Duck	Rabbit	Recipe
D1	0	0	0.097	0	0
D2	0.199	0.199	0.097	0	0
D3	0	0	0.097	0.199	0.111
D4	0	0	0	0.398	0.222
D5	0.398	0.398	0.097	0	0.222
Query	**1*.398**	**0**	**1*.097**	**0**	**1*.222**

Word	IDF
Beijing	.398
Dish	.398
Duck	.097
Rabbit	.398
Recipe	.222

datasciencedojo
unleash the data scientist in you

TF-IDF Search Exercise

- Cosine similarity of query and each doc
- $D1 = [0, 0, 0.097, 0, 0]$
- $Q = [0.398, 0, 0.097, 0, 0.222]$
- $\cos(D1, Q) = \frac{0*0.398+0*0+0.097*0.097+0*0+0*0.222}{\sqrt{0.097^2}*\sqrt{0.398^2+0.097^2+0.222^2}}$
- $\cos(D1, Q) = \frac{0.00941}{0.0452} = 0.208$

datasciencedojo
unleash the data scientist in you

Cosine similarities

	Beijing	Dish	Duck	Rabbit	Recipe	Cos(D,Q)
D1	0	0	0.097	0	0	0.208
D2	0.199	0.199	0.097	0	0	0.639
D3	0	0	0.097	0.199	0.111	0.256
D4	0	0	0	0.398	0.222	0.232
D5	0.398	0.398	0.097	0	0.222	0.760
Query	**.398**	**0**	**.097**	**0**	**.222**	1

datasciencedojo
unleash the data scientist in you

Final ordered list

- **D5** = "Last week Li has shown you how to make the Sechuan duck. Today we'll be making Chinese dumplings (Jiaozi), a popular dish that I had a chance to try last summer in Beijing. There are many recipies for Jiaozi."
- **D2** = "Beijing Duck is mostly prized for the thin, crispy duck skin with authentic versions of the dish serving mostly the skin."
- **D3** = "Bugs' ascension to stardom also prompted the Warner animators to recast Daffy Duck as the rabbit's rival, intensely jealous and determined to steal back the spotlight while Bugs remained indifferent to the duck's jealousy, or used it to his advantage. This turned out to be the recipe for the success of the duo."
- **D4** = "6:25 PM 1/7/2007 blog entry: I found this great recipe for Rabbit Braised in Wine on cookingforengineers.com."
- **D1** = "If it walks like a duck and quacks like a duck, it must be a duck."

datasciencedojo
unleash the data scientist in you

Recommending items

- Advanced Methods
  - Classification algorithms
    - Predict either ratings or like/dislike
  - Information retrieval techniques
    - Well studied field, wide diversity of models

Content-based recommenders

Advantages

- **No community required**
  - Only need the items and a single user profile for recommendation.
- **Transparency**
  - CB models can tell you why they recommend an item, not subject to vagaries of human taste
- **Good cold start**
  - New items can be suggested before being rated by a substantial number of users.

Content-based recommenders

Disadvantages

- **Limited content analysis**
  - Requires well annotated content for good recommendations.
- **Over-specialization**
  - Users will tend to be recommended items very similar to what they have enjoyed in the past
  - Very limited discoverability
- **New users**
  - Limited user information results in bad recommendations.

Overview

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

Evaluating Recommendation

- Error Rate Metrics
  - **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

- Order matters, not exact ranking value
- Graded Relevance
  - Have humans assign scores to possible results
  - Ideal results will be ordered by relevance, high to low
- Discounted cumulative gain (DCG)
  - Logarithmic reduction factor

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

Where:

- *pos* is the length of the recommendation list
- rel_i returns the relevance of recommendation at position *i*

datasciencedojo
unleash the data scientist in you

Metrics

- **Ideal discounted cumulative gain (IDCG)**
  - DCG value when items are ordered perfectly

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

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

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

  - Normalized to the interval [0..1]

datasciencedojo
unleash the data scientist in you

QUESTIONS

Overview

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

datasciencedojo
unleash the data scientist in you