Web Data Mining

Lecture 10: Recommender Systems

Jaroslav Kuchař & Milan Dojčinovski
jaroslav.kuchar@fit.cvut.cz, milan.dojchinovski@fit.cvut.cz

Czech Technical University in Prague - Faculty of Information Technologies - Software and Web Engineering

CC BY SA

OPPA EVROPSKÁ UNIE

Summer semester 2019/2020
Humla v0.3

Overview

- Introduction
- Collaborative Filtering
- Content Based Filtering
- Other Recommender Systems
- Other Aspects
- Supplementary Material

Motivation

- Too much information
  - *users are overloaded with information*
  - *many choices available*
- Examples
  - *Thousands of news articles and blog posts every day*
  - *Milions of movies, books, sound tracks*
  - *Many restaurants, hotels, apps, people, ...*
- People can't assess every piece of information!
  - *Need for an intelligent information delivery!*
- Solution
  - ***Recommender Systems***
- Relation to Information Retrieval
  - *Many similar approaches on the background*
  - *Goals*
    - → *IR - "I know what I am looking for"*
    - → *RS - "I am not sure what i am looking for"*

Recommender Systems

- Systems for recommending items to users
  - *books, movies, web pages, songs, ...*
  - *based on users preferences in the past*
- Enhance user experience
  - *assist users in finding information*
  - *reduce search and navigation time*
- Recommendations are on first place on the internet
  - *Google's recommendations*
    - → *web pages, advertisments*
    - → *recommendations generate 38% more clicks*
  - *Youtube's recommendations*
    - → *videos, channels to subscribe*
  - *Netflix's recommendations*
    - → *movies*
    - → *2/3 watched movies are recommended*
  - *Facebook's recommendations*
    - → *friends, posts, advertisments*
  - *Twitter's recommendations*
    - → *tweets, people to follow*
  - *Amazon's recommendations*
    - → *books, DVDs, electronics, toys, ...*
    - → *35% sales from recommendations*

Information Used for Recommendations

- Used information can come from different sources:
  - *browsing and searching - past behavior*
    $\rightarrow$ *e.g. clicks, logs*
  - *purchase data*
    $\rightarrow$ *e.g. bought book or movie*
  - *explicit feedback provided by the users*
    $\rightarrow$ *e.g. likes, rating stars*
  - *comments on products*
    $\rightarrow$ *requires opinion mining*
  - *demographic data*
  - *context*
  - *relations to other users*
  - *item similarities*
  - ...
- Recommender systems usually exploit information from more sources

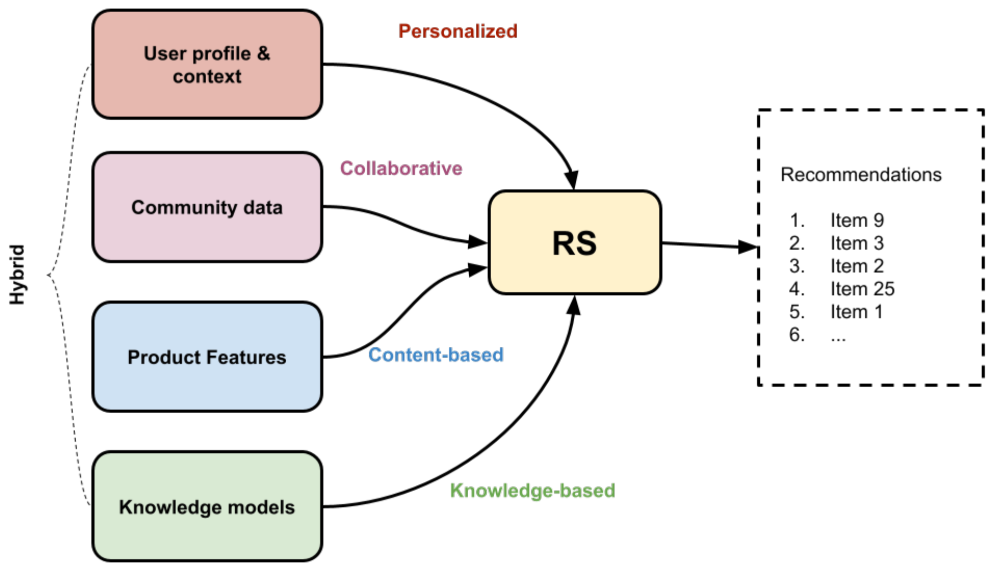
Personalization

- Adapting to the individual needs
- Accommodate the differences between individuals according to their preferences, needs or interests
- Key idea in recommendation systems
- Assumptions
  - *if user A and user B are similar, then in the future user A might be interested in items from user B*
    $\rightarrow$ *recommend items from similar users*
  - *if user A is interested in item I, then in the future user A might be interested in items similar to item I*
    $\rightarrow$ *recommend items similar to the items in the basket*

Types of Recommender Systems

- Collaborative filtering
  - *analyze the user-item interactions for each user, and recommend items from similar users*
- Content-based filtering
  - *analyze the content of the items (bought books) and recommend similar content (similar books)*
- Knowledge-based
  - *In situation when limited information is available*
  - *Uses knowledge-base*
  - *which item should be recommended in which context?*
- Other - demographic, …
- Hybrid
  - *combination of approaches*

Types of Recommender Systems (cont.)



Overview

- Introduction
- Collaborative Filtering
- Content Based Filtering
- Other Recommender Systems
- Other Aspects
- Supplementary Material

Collaborative Filtering

- Assumption
  - *User with similar taste in past will have similar taste in future*
- Analyze user-item interactions
  - *likes, watched video, bought book*
  - *do not analyze the content of the items being recommended*
- Method
  - *Produce recommendations by computing the similarity between a user's preferences and preferences of other people*
  - *Suggest new items to users who have similar preferences with others*
- Approaches
  - *User-based - find similar users to me and recommend what they liked*
  - *Item-based - find similar items to those that I have previously liked*
- Widely used in practice

CF Inputs

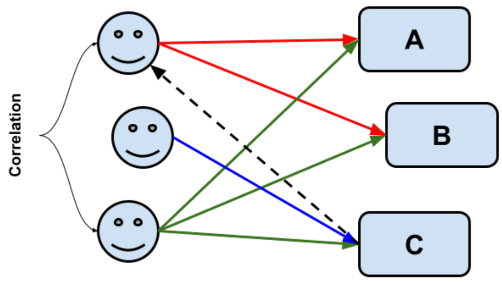
- General lists
  - *List of users*
  - *List of items*
- Associations
  - *Each user has associations to items*
    - → *Explicit*
      - → *ratings*
    - → *Implicit*
      - → *purchases*
      - → *listen records*
- Metrics/methods
  - *to measure similarity between users*
  - *to select top neighbors*
- Active user

Collaborative Filtering steps

- Steps
  - *Identify relevant input data*
    - → *Collect user-item interactions for the users*
      - → *ratings (0-5 stars), likes, etc*
  - *Identify set of users most similar to the active user - neighborhood*
    - → *Compute user-to-user similarity*
      - → *based on their interactions with the items*
  - *Identify items for these similar user*
  - *Generate prediction*
    - → *that would be given by the active user*
  - *Recommend items*
- Main challenge
  - ***how to compute the similarity between the users***

User-based CF

- Steps
  - *For the active user find top similar users*
    → neighborhood
  - *Detect ratings of neighbors for unrated items*
  - *Compute average of ratings*
  - *Recommend items with highest predicted ratings*
- In short
  - *"You may like it because other liked it"*



Example

- Ratings of 5 users for 6 items
- What can we recommend for the user 3?

users\items	1	2	3	4	5	6	mean rating	Cosine(u, u3)	Pearson (u, 3)
u1	7	6	7	4	5	4	5.5	0.956	0.894
u2	6	7		4	3	4	4.8	0.981	0.939
u3		3	3	1	1		2	1.0	1.0
u4	1	2	2	3	3	4	2.5	0.789	-1.0
u5	1		1	2	3	3	2	0.645	-0.817

- Descriptions:
  - *For user 3 we can recommend missing items*
    → Possibilities - item 1 and 6
  - *User 1 and user 2 are the most similar*

Similarities

- Cosine similarity
  - *Defined as cosine of the angle between the vectors*

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

- Cosine similarity in action

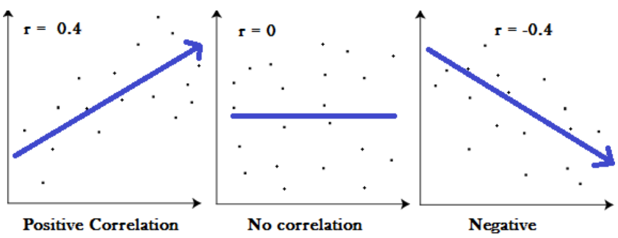
users\items	1	2	3	4	5	6	mean rating	Cosine(u, u3)	Pearson (u, 3)
u1	7	6	7	4	5	4	5.5	0.956	0.894
u3		3	3	1	1		2	1.0	1.0

- cosine(user1, user3)
  - $A * B = 6 * 3 + 7 * 3 + 4 * 1 + 5 * 1$
  - $||A|| * ||B|| = \sqrt{6^2 + 7^2 + 4^2 + 5^2} * \sqrt{3^2 + 3^2 + 1^2 + 1^2}$
  - *cosine(user1, user3) = A * B / ||A|| * ||B|| = 0.956*

Similarities (cont.)

- Pearson Similarity Metric
  - *Computes how correlated are two vectors*
  - *Similarity in range [1,-1]*
    - → *closer to 1 items are correlated, closer to -1 items are negatively correlated*
    - → *negative value means as one variable increases the other decreases*
    - → *positive value means as one variable increases also the other increases*
    - → *in our example, users tends to rank photos with similar ratings*

$$corr(i,j) = \frac{\sum_{i=u}^{U}(R_{ui} - R_i)(R_{uj} - R_j)}{\sqrt{\sum_{i=u}^{U}(R_{ui} - R_i)^2}\sqrt{\sum_{i=u}^{U}(R_{uj} - R_j)^2}}$$



Similarities (cont.)

- Pearson Similarity Metric in Action

users\items	1	2	3	4	5	6	mean rating	Cosine(u, u3)	Pearson (u, 3)
u1	7	6	7	4	5	4	5.5	0.956	0.894
u3		3	3	1	1		2	1.0	1.0

- corr(user1, user3)
  - $A = (6 - 5.5) * (3 - 2) + (7 - 5.5) * (3 - 2) \ldots$
  - $B = \sqrt{0.5^2 + 1.5^2 + \ldots} * \sqrt{1^2 + 1^2 \ldots}$
  - $corr(user1, user3) = A / B = 0.894$

User-based Recommendation

- How to use the user-user similarity information and predict rating for an item?
- Pearson-weighted average of ratings
  - *predicts rating of missing items*
  - *predicted ratings are used for recommendation of items*
- Raw rating prediction

$$pred(a, b) = \frac{\sum_{b \in N} sim(a,b) \,.\, r_b}{\sum_{b \in N} sim(a,b)}$$

- Mean-centered prediction

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

User-based Recommendation (cont.)

- What can we recommend for the user 3?

users\items	1	2	3	4	5	6	mean rating	Cosine(u, u3)	Pearson (u, 3)
u1	7	6	7	4	5	4	5.5	0.956	0.894
u2	6	7		4	3	4	4.8	0.981	0.939
u3		3	3	1	1		2	1.0	1.0
u4	1	2	2	3	3	4	2.5	0.789	-1.0
u5	1		1	2	3	3	2	0.645	-0.817

- In action
  - *by selecting top-2 similar users*

$$\rightarrow user3,\ item1 = \frac{7 * 0.894 + 6 * 0.939}{0.894 + 0.939} = 6.49$$

$$\rightarrow user3,\ item6 = \frac{4 * 0.894 + 4 * 0.939}{0.894 + 0.939} = 4$$

Item-based CF

- Computes similarity between items (columns in the rating matrix)
  - *Use this similarity to predict ratings*
  - *More computationally efficient*
    - *→ number of items << number of user*
  - *Similar similarity metrics can be used*
  - *Prediction is computed in the same way as in the user-based*

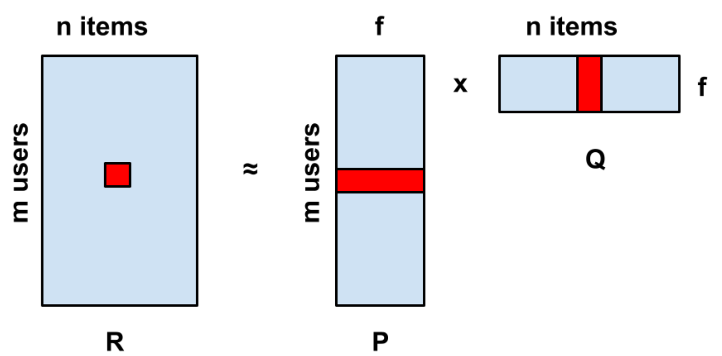
users\items	1	2	3	4	5	6
u1	7	6	7	4	5	4
u2	6	7		4	3	4
u3		3	3	1	1	
u4	1	2	2	3	3	4
u5	1		1	2	3	3
sim(1, i)	1	0.735	0.912	-0.848	-0.813	-0.990
sim(6, i)	-0.990	-0.622	-0.912	0.829	0.730	1

Matrix Factorization CF

- Model-based recommendation
- Main idea
  - *Discover latent factors of users or items*
  - *Use those latent factors for recommendations*
  - *Assumption: Ratings can be inferred from a model put together from a smaller number of parameters*
- Latent factors
  - *features that describe recommended items o users*
  - *e.g. categories of movies*
  - *can be discovered automatically*
- Approaches
  - *Singular Value Decomposition (SVD)*
  - *Matrix Factorization*

Matrix Factorization CF (cont.)

- Basic Matrix Factorization



- Description
  - *R - rating matrix (m users, n items)*
  - *P - user features matrix (m users, f features/latent factors)*
  - *W - item features matrix (n items, f features/latent factors)*
  - *Rating cane computed as*
    - $\rightarrow r_{ui} \approx q_i^T \times p_u$

Matrix Factorization CF - Example

- Rating matrix

users\items	item1	item2	item3
u1	5	3	4
u2	?	2	4
u3	4	2	?

- P

users\features	f1	f2	f3	f4	f5
u1	0.276	-0.377	-1.262	-1.548	0.473
u2	0.399	-0.527	-0.289	-1.516	0.737
u3	0.225	-0.291	-1.067	-1.225	0.374

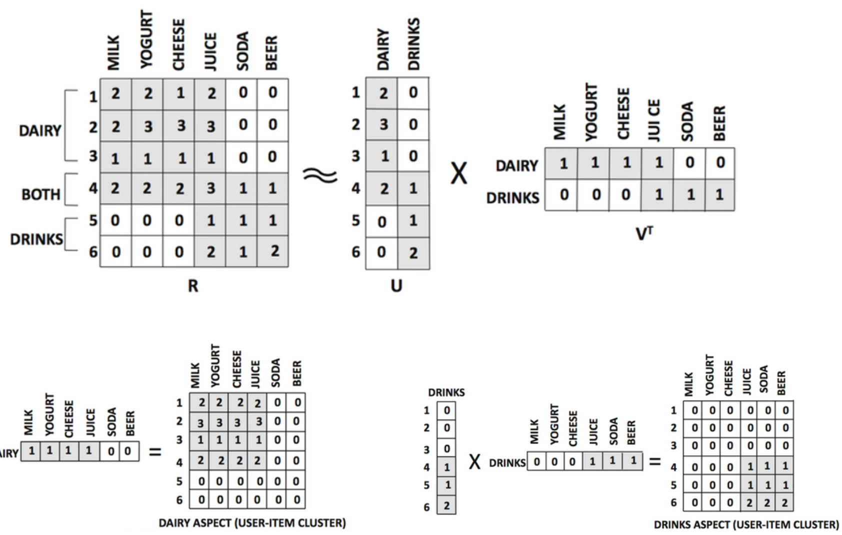
- W

items\features	item1	item2	item3
f1	0.302	0.143	0.434
f2	-0.405	-0.202	-0.575
f3	-1.399	-0.923	-0.468
f4	-1.678	-0.940	-1.720
f5	0.512	0.239	0.796

- Rating prediction: $r(user3, item3) = P_3 f \times W_f 3 = 3.168$

Matrix Factorization CF - Example 2

- Product Ratings



Charu C. Aggarwal: Recommender Systems - The Textbook. Springer 2016, ISBN 978-3-319-29657-9, pp. 1-498.

Matrix Factorization CF - Example 3

- *Movie Ratings*

User ID	Top Gun	Rambo	Young Guns	Lonesome Dove
1	5	4	1	1
2	4	5	1	1
3	1	1	5	5
4	1	1	4	5
5	1	1	5	4

- *Feature Vectors for movies*

	Feature 1	Feature 2
	Military	**Western**
Top Gun	3.836413	0.5803331
Rambo	7.744796	0.3196278
Young Guns	1.217036	7.829875
Lonesome Dove	1.120779	9.8565764

* "Nonnegative Matrix Factorization and Recommendor Systems." R-bloggers. N.p., 24 Oct. 2012. Web. 28 Apr. 2017.

Problems with Collaborative Filtering

- Cold start
  – *needs to have enough other users already in the system to find 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
  – *new not rated items are not recommended*
- Popularity bias
  – *can not recommend items to someone with unique tastes*
  – *specific items are always popular*
- Lack of Transparency
  – *recommendations can not be easily explained, why an item was recommended*

Overview

- Introduction
- Collaborative Filtering
- Content Based Filtering
- Other Recommender Systems
- Other Aspects
- Supplementary Material

Content Based Filtering

- Produce recommendations by analyzing the content (metadata) of the items
  – *metadata - categories, tags, textual content, authors, etc.*
  – *automatically extracted fetures - e.g. text mining*
- Approach
  – *Build representation for each item*
    → *Represent items usually as a vector*
  – *Content-based learning of user profiles*
    → *Can be represented in many ways*
      → *similar vector as for the items*
      → *set of keywords*
      → *classification or regression models*
  – *Filtering and recommendations*
    → *Identify candidates*
      → *Measure the similarity between the user profile vector and each item vector*
      → *Classify items using user profiles as classifiers*
    → *Recommend top-N most relevant items to the user*

Content Based Filtering - Example

- Items

	Action	Sci-fi	Comedy	Romance	Sport	Drama
item1	1	1	0	0	0	0
item2	1	0	1	0	0	0
item3	0	0	1	1	0	0
item4	0	0	0	0	1	1
item5	0	0	1	0	1	0
item6	0	1	0	0	0	1
item7	1	1	1	0	0	0

- User profile

	Action	Sci-fi	Comedy	Romance	Sport	Drama
user1	1	0	0	0	0	1

- Similarity measures
  - *item1, item2, item4, item6, …*

Content Based Filtering - Example 2

- Rule-based classifier

song\cat.	Drums	Guitar	Beat	Classical	Symphony	Orchestra	L/D
1	1	1	1	0	0	0	D
2	1	1	0	0	0	1	D
3	0	1	1	0	0	0	D
4	0	0	0	1	1	1	L
5	0	1	0	1	0	1	L
6	0	0	0	1	1	0	L
t1	0	0	0	1	0	0	?
t2	1	0	1	0	0	0	?

- Rules (supp=0.3, conf=0.75)
  - *Rule 1: {Classical} -> Like (50%, 100%)*
  - *Rule 2: {Symphony} -> Like (33%, 100%)*
  - *Rule 3: {Classical, Symphony} -> Like (33%, 100%)*
  - *Rule 4: {Drums, Guitar} -> Dislike (33%, 100%)*
  - *Rule 5: {Drums} -> Dislike (33%, 100%)*
  - *Rule 6: {Beat} -> Dislike (33%, 100%)*
  - *Rule 7: {Guitar} -> Dislike (50%, 75%)*
- Classifications
  - *t1 – Rule 1 – Like*
  - *t2 – Rule 5, Rule 6 – Dislike*

Advantages of Content Based Filtering

- No need for data from other users
  - *No cold start or sparsity problems*
- Tastes of unique users are met
- Can recommend new and unpopular items
  - *First rater problem is solved*
- Can provide explanation why an item was recommended
  - *by listing content features that caused an item to be recommended*

Disadvantages of Content Based Filtering

- Content should be encoded as meaningful features
- Users tastes must be possible to represent as vectors
- Problems with new users with no profile
- Does not exploit the collective intelligence information
  - *how other users interacted with the item*
- Difficult to implement serendipity
- Easy to overfit

Overview

- Introduction
- Collaborative Filtering
- Content Based Filtering
- Other Recommender Systems
- Other Aspects
- Supplementary Material

Knowledge-Based RS

- Applications
  – *expensive items, not frequently purchased, few ratings*
  – *time span important (e.g., technological products)*
    $\rightarrow$ *ratings may be time-sensitive*
    $\rightarrow$ *"The ratings on an old car or computer are not very useful for recommendations because they evolve with changing product availability"*
  – *explicit requirements of user*
    $\rightarrow$ *interactivity is a crucial component of such systems*
- Limitations of other approaches
  – *collaborative filtering unusable – not enough data*
  – *content based – similarity not sufficient*
- Requires functional/domain knowledge
  – *about how a particular item meets a particular user need*

Knowledge-Based RS (cont.)

- Types
  - *Constraint-based recommender systems*
    - *→ users typically specify requirements or constraints (e.g., lower or upper limits) on the item attributes*
    - *→ domain-specific rules are used to match the user requirements*
      - *→ e.g. "Cars before year 1970 do not have cruise control."*
      - *→ e.g. "Older investors do not invest in ultrahigh-risk products."*
    - *→ process is interactively repeated until the user arrives at her desired results*
  - *Case-based recommender systems*
    - *→ specific cases are specified by the user as targets*
    - *→ similarity metrics are defined on the item attributes to retrieve similar items to these targets*
    - *→ returned results are often used as new target cases*
    - *→ interactive process is used to guide the user towards the final recommendation*
- Interactions
  - *Conversational systems*
    - *→ user preferences are determined in the context of a feedback loop*
  - *Search-based systems*
    - *→ a preset sequence of questions*
  - *Navigation-based recommendation*

Constraint-Based RS - Example

- Buying homes

Item	Beds	Baths	Locality	Type	Floor Area	Price
1	3	2	Bronx	Townhouse	1600	220,000
2	5	2.5	Chappaqua	Split-level	3600	973,000
3	4	2	Yorktown	Ranch	2600	630,000
4	2	1.5	Yorktown	Condo	1500	220,000
5	4	2	Ossining	Colonial	2700	430,000

- Customer-specified attributes
  - *Marital-status (categorical), Family-Size (numerical), suburban-or-rural (binary), Min-Bedrooms (numerical), Max-Bedrooms (numerical), Max-Price (numerical)*
- Domain knowledge for mapping of requirement into the attributes
  - *Suburban-or-rural=Suburban -> Locality= {List of relevant localities}*
  - *Marital-status=single -> Min-Bedrooms<=5*
  - *Family-Size>=5 -> Min-Bedrooms<=3*
  - *Min-Bedrooms>=3 -> Price>=100,000*

Other RS

- Context-sensitive RS
  - *context - additional information that defines the specific situation under which recommendations are made*
    - → *Time - morning, evening, weekdays, weekends, holidays*
    - → *Location - recommendation for a restaurant in his locality*
    - → *Social information - e.g. social circles can affect the recommendation process*
- Ensemble or Hybrid RS
  - *Weighted*
    - → *scores of several recommender systems are combined into a single unified score*
  - *Switching*
    - → *switches between various recommender systems depending on current needs*
  - *Cascade*
    - → *one recommender system refines the recommendations given by another*
  - *Feature augmentation*
    - → *the output of one recommender system is used to create input features for the next*
  - *Mixed*

Overview

- Introduction
- Collaborative Filtering
- Content Based Filtering
- Other Recommender Systems
- Other Aspects
- Supplementary Material

Evaluation of RS

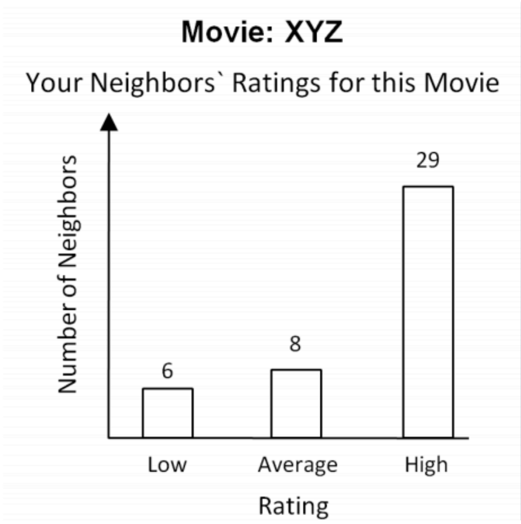
- Evaluations - difficult to properly measure the quality
  - *many approaches*
  - *many metrics*
- Approaches
  - *off-line*
    - → *simulations of users and their behavior*
    - → *issues with overfitting*
    - → *the goal is to filter out inappropriate approaches*
  - *user-studies*
    - → *small group of real users*
    - → *expensive*
  - *on-line*
    - → *AB testing*
    - → *can be risky - user dissatisfaction*
- Metrics
  - *Root Mean Squared Error (RMSE) - normalized and averaged*
  - *Precision, Recall (at N)*
  - *F-measure, Area Under the ROC Curve (AUC)*
  - *Normalized Cumulative Discounted Gain (NDCG)*
  - *Click-through rate (CTR)*
  - *Coverage - Novelty, Serendipity, Diversity*

Explanations

- Situation
  - *recommendations: selection (ranked list) of items*
  - *explanations: (some) reasons for the choice*
- Why explanations?
  - *transparency, trustworthiness, validity, satisfaction (users are more likely to use the system)*
  - *persuasiveness (users are more likely to follow recommendations)*
  - *effectiveness, efficiency (users can make better/faster decisions)*
  - *education (users understand better the behaviour of the system, may use it in better ways)*
- Examples
  - *knowledge-based recommender*
    - → *"Because you, as a customer, told us that simple handling of car is important to you, we included a special sensor system in our offer that will help you park your car easily."*
  - *recommendations based on item-similarity*
    - → *"Because you watched X we recommend Y"*

Explanations - Example

- Collaborative filtering



Movie: XYZ

Personalized Prediction: ****

Your Neighbors` Ratings for this Movie

Rating	Number of Neighbors
★	2
★★	4
★★★	8
★★★★	20
★★★★★	9

* Explaining collaborative filtering recommendations, Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW

Attacks

- Most vulnerable
  – *Collaborative filtering RS*
- Reasons for attacks
  – *make the system worse (unusable)*
  – *influence rating (recommendations) of a particular item*
    → *push attacks – improve rating of "my" items*
    → *nuke attacks – decrease rating of "opponent's" items*
- Approach
  – *creating a set of fake feedbacks from many different users*
    → *shilling attack - users become shills in the attack*

Attacks (cont.)

- Example

Items

Users	1	2	3	4	5	6	7	
a	+	−		+	+		+	Authentic profiles
b	−	+	+	−	−		−	
c	+	−	+		−	−	−	
d	−	+	+	−				
e	−		−	−	−		−	
f	+	−	+	+	+		+	
g		−	+	+	−	−	+	
h	+	−	+	+	+		?	Target profile
i	+	−	+		−	−	−	Attack profiles
j	−	+	+	−			−	
k	−		−	−	−		−	
l	+	−	+	+	+		−	
m		−	+	+	−	−	−	

Attacks - Types

- Levels
  - *more knowledge about system -> more efficient attack*
- Types
  - *random attack*
    - → *generate profiles with random values*
    - → *requires global mean*
  - *average attack*
    - → *effective attack on memory-based systems (average ratings -> many neighbors)*
    - → *requires global mean and mean of each item*
  - *bandwagon attack*
    - → *high rating for "blockbusters", random values for others*
    - → *requires to know which items are the most popular*
  - *segment attack*
    - → *insert ratings only for items from specific segment*
    - → *attack item-based collaborative filtering*
  - *love/hate attack*
    - → *the nuked item is set to the minimum rating value , whereas the other items are set to the maximum*
  - *reverse bandwagon*
    - → *widely disliked items are used as filler items to mount the attack*
  - *probe attack*
    - → *a seed profile is created by the attacker, and the predictions generated by the recommender system are used to learn related items and their ratings*

Attacks - Detection

- Attack detections
  - *Unsupervised attack detection algorithms*
    - *→ ad hoc rules are used to detect fake profiles*
    - *→ to identify the key characteristics of attack profiles that are not similar to genuine profiles*
      - *→ if a profile (or significant portion of it) is identical to many other profiles*
      - *→ Number of prediction differences, Degree of disagreement with other users, Rating deviation from mean agreement, Degree of similarity with top-k neighbors*
  - *Supervised attack detection algorithms*
    - *→ classification models to detect attacks*
      - *→ number of profiles to which a given user profile is identical can be used as a feature for that user profile*
- Robust recommender systems
  - *preventing automated attacks with CAPTCHAs*
  - *detections using metrics, clusterings, neighborhoods, more complex algorithms etc.*

Overview

- Introduction
- Collaborative Filtering
- Content Based Filtering
- Other Recommender Systems
- Other Aspects
- Supplementary Material

Euclidean Based Proximity Measure

- Represent query and document in a vector space model
  - *represent a query as a vector*
  - *represent a document as a vector*
- Compute the distance of the two vectors in a multi-dimensional space
- An Eucledian distance of two vectors

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$

- Eucledian distance is a bad idea
  - *if a query term occurs more times in the document, the distance will be larger*
  - *distance is large, altough the distribution of terms are similar*

Cosine Based Proximity Measure

- Use angle instead of distance
  - *compute cosine of the angle between the query and document vectors*
- Cosine similarity of two vectors (query vs. document)

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

- Long and short vectors now have comparable weights
- If two documents are identical, then the angle is 0
  - *cosine of 0 angle = 1*
  - *similarity of two identical vectors is 1*
- Rank the documents according to the angle of the query

Precision and Recall

- Precision
  – *fraction of retrieved documents that are relevant*
  – P(relevant|retrieved)
- Recall
  – *fraction of relevant documents that are retrieved*
  – P(retrieved|relevant)

Contingency table:

/	Relevant	Non-relevant
Retrieved	tp (true positive)	fp (false positive)
Not retrieved	fn (false negative)	tn (true negative)

Precision and Recall

/	Relevant	Non-relevant
Retrieved	tp (true positive)	fp (false positive)
Not retrieved	fn (false negative)	tn (true negative)

- tp - true positive
  – *relevant documents that are retrieved*
- fp - false positive
  – *non-relevant documents that are retrieved*
- fn - false negative
  – *non-retrieved relevant documents*
- tn - true negative
  – *non-retrieved non-relevant documents*

Precision and Recall Measure

/	Relevant	Non-relevant
Retrieved	tp (true positive)	fp (false positive)
Not retrieved	fn (false negative)	tn (true negative)

- Precision
  - $P = \frac{tp}{(tp+fp)}$
- Recall
  - $R = \frac{tp}{(tp+fn)}$

Accuracy

- Assume
  - *an engine for a given query classifies each document as "relevant" or "non relevant"*
- The accuracy
  - *the fraction of these classifications that are correct*
  - $\frac{(tp+tn)}{(tp+fp+fn+tn)}$
- Extremely skewed
  - *normally over 99.9% of the documents are in the non-relevant category*
- Widly used measure in evaluating machine learning systems

F-Measure

- Also known as F-score or F1 score
- Assesses precision/recall tradeoff
  - $F = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$