Principles/Social Media Mining

CIS 600

Week 12: Recommender Systems

Edmund Yu, PhD
Associate Teaching Professor
esyu@syr.edu

Recommender system

From Wikipedia, the free encyclopedia

A **recommender system** or a **recommendation system** (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.[1][2]

Recommender systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts,[3] collaborators,[4] jokes, restaurants, garments, financial services,[5] life insurance, romantic partners (online dating), and Twitter pages.[6]

Recommender systems

Concepts
Collective intelligence · Relevance · Star ratings · Long tail

Methods and challenges
Cold start · Collaborative filtering · Dimensionality reduction · Implicit data collection · Item-item collaborative filtering · Preference elicitation · Similarity search · Social Loafing

Implementations
Collaborative search engine · Content discovery platform · Decision support system · Music Genome Project · Product finder

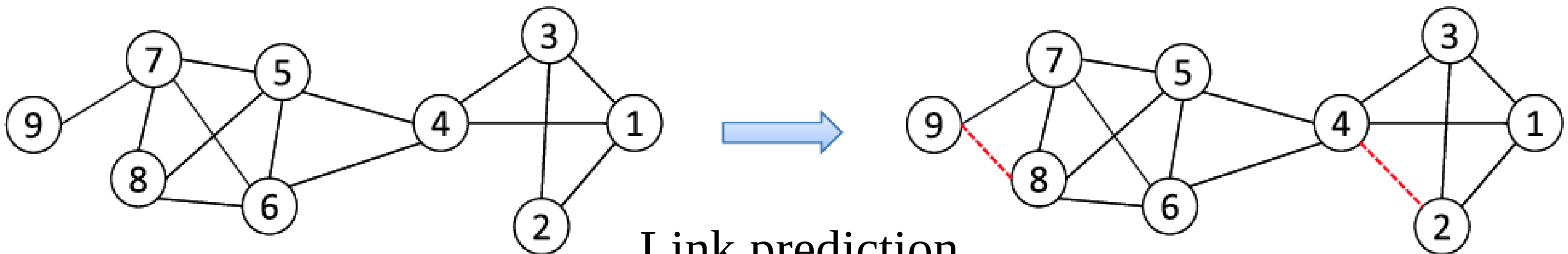
Research
GroupLens Research · MovieLens · Netflix Prize

V · T · E

Contents [hide]

Area 5: Recommendation

- ❖ Very common in social media applications
  - ❖ Tag, Friend, Group, Media, Link Recommendations

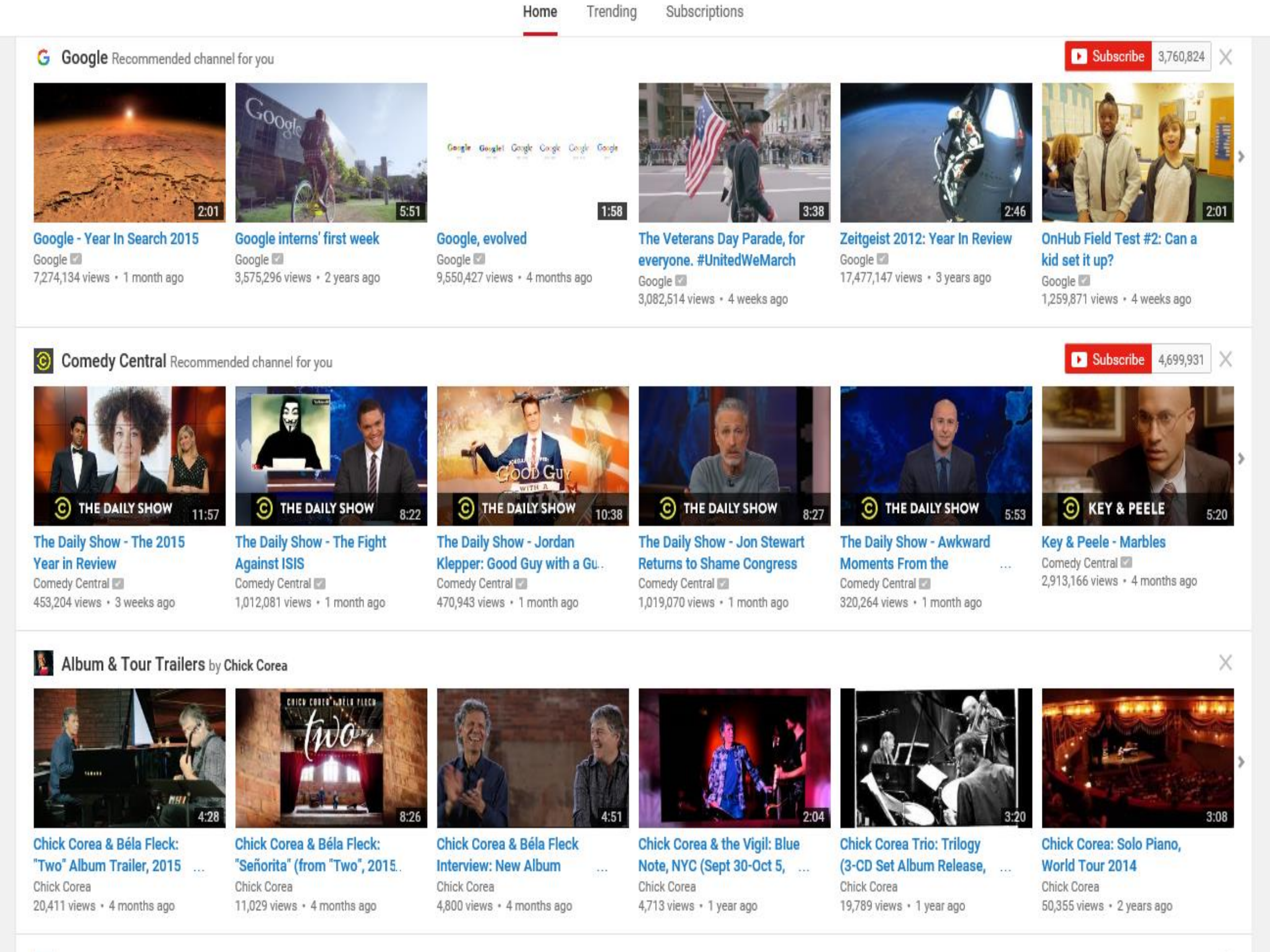


Link prediction

Recommender Systems: Facebook

- Facebook friend recommendations
- "People you may know"
- "Based on mutual friends, work and education information, networks you're part of, contacts and many other factors."
- "Since our formula is automatic, you might occasionally see people you don't know or don't want to be friends with. To remove them from view, just click the X next to their names."

http://www.facebook.com/help/?page=199421896769556



Home
Trending
Subscriptions
Google Recommended channel for you
Subscribe 3,760,824
2:01
Google - Year In Search 2015
Google
7,274,134 views • 1 month ago
5:51
Google interns' first week
Google
3,575,296 views • 2 years ago
1:58
Google, evolved
Google
9,550,427 views • 4 months ago
3:38
The Veterans Day Parade, for everyone. #UnitedWeMarch
Google
3,082,514 views • 4 weeks ago
2:46
Zeitgeist 2012: Year In Review
Google
17,477,147 views • 3 years ago
2:01
OnHub Field Test #2: Can a kid set it up?
Google
1,259,871 views • 4 weeks ago
Comedy Central Recommended channel for you
Subscribe 4,699,931
THE DAILY SHOW 11:57
The Daily Show - The 2015 Year in Review
Comedy Central
453,204 views • 3 weeks ago
THE DAILY SHOW 8:22
The Daily Show - The Fight Against ISIS
Comedy Central
1,012,081 views • 1 month ago
THE DAILY SHOW 10:38
The Daily Show - Jordan Klepper: Good Guy with a Gu...
Comedy Central
470,943 views • 1 month ago
THE DAILY SHOW 8:27
The Daily Show - Jon Stewart Returns to Shame Congress
Comedy Central
1,019,070 views • 1 month ago
THE DAILY SHOW 5:53
The Daily Show - Awkward Moments From the ...
Comedy Central
320,264 views • 1 month ago
KEY & PEELE 5:20
Key & Peele - Marbles
Comedy Central
2,913,166 views • 4 months ago
Album & Tour Trailers by Chick Corea
4:28
Chick Corea & Béla Fleck: "Two" Album Trailer, 2015 ...
Chick Corea
20,411 views • 4 months ago
8:26
Chick Corea & Béla Fleck: "Señorita" (from "Two", 2015...
Chick Corea
11,029 views • 4 months ago
4:51
Chick Corea & Béla Fleck Interview: New Album ...
Chick Corea
4,800 views • 4 months ago
2:04
Chick Corea & the Vigil: Blue Note, NYC (Sept 30-Oct 5, ...
Chick Corea
4,713 views • 1 year ago
3:20
Chick Corea Trio: Trilogy (3-CD Set Album Release, ...
Chick Corea
19,789 views • 1 year ago
3:08
Chick Corea: Solo Piano, World Tour 2014
Chick Corea
50,355 views • 2 years ago

Recommender Systems: Netflix

- ❖ Netflix (movie recommendation)
- ❖ $1M prize for 10% accuracy improvement

Netflix Prize

From Wikipedia, the free encyclopedia

The **Netflix Prize** was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films, i.e. without the users or the films being identified except by numbers assigned for the contest.

The competition was held by Netflix, an online DVD-rental and video streaming service, and was open to anyone who is neither connected with Netflix (current and former employees, agents, close relatives of Netflix employees, etc.) nor a resident of certain blocked countries (such as Cuba or North Korea).[1] On September 21, 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%.[2]

Contents [hide]

Recommender systems

Concepts
Collective intelligence · Relevance · Star ratings · Long tail

Methods and challenges
Cold start · Collaborative filtering · Dimensionality reduction · Implicit data collection · Item-item collaborative filtering · Matrix factorization · Preference elicitation · Similarity search

Implementations
Collaborative search engine · Content discovery platform · Decision support system · Music Genome Project · Product finder

Research
GroupLens Research · MovieLens · **Netflix Prize**

V · T · E

days, until July 26, 2009 18:42:37 UTC, to make submissions that will be considered for this Prize.[19]

On July 25, 2009 the team "The Ensemble", a merger of the teams "Grand Prize Team" and "Opera Solutions and Vandelay United", achieved a 10.09% improvement over Cinematch (a Quiz RMSE of 0.8554).[20][21]

On July 26, 2009, Netflix stopped gathering submissions for the Netflix Prize contest.[22]

The final standing of the Leaderboard at that time showed that two teams met the minimum requirements for the Grand Prize. "The Ensemble" with a 10.10% improvement over Cinematch on the Qualifying set (a Quiz RMSE of 0.8553), and "BellKor's Pragmatic Chaos" with a 10.09% improvement over Cinematch on the Qualifying set (a Quiz RMSE of 0.8554).[23] The Grand Prize winner was to be the one with the better performance on the Test set.

On September 18, 2009, Netflix announced team "BellKor's Pragmatic Chaos" as the prize winner (a Test RMSE of 0.8567), and the prize was awarded to the team in a ceremony on September 21, 2009.[24] "The Ensemble" team had matched BellKor's result, but since BellKor submitted their results 20 minutes earlier, the rules award the prize to BellKor.[21][25]

The joint-team "BellKor's Pragmatic Chaos" consisted of two Austrian researchers from Commendo Research & Consulting GmbH, Andreas Töscher and Michael Jahrer (originally team BigChaos), two researchers from AT&T Labs, Robert Bell, and Chris Volinsky, Yehuda Koren from Yahoo! (originally team BellKor) and two researchers from Pragmatic Theory, Martin Piotte and Martin Chabbert.[26] As required, they published a description of their algorithm.[27]

The team reported to have achieved the "dubious honors" (*sic* Netflix) of the worst RMSEs on the *Quiz* and *Test* data sets from among the 44,014 submissions made by 5,169 teams was "Lanterne Rouge", led by J.M. Linacre, who was also a member of "The Ensemble" team.

Cancelled sequel [edit]

On March 12, 2010, Netflix announced that it would not pursue a second Prize competition that it had announced the previous August. The decision was in response to a lawsuit and Federal Trade Commission privacy concerns.[28]

Privacy concerns [edit]

Although the data sets were constructed to preserve customer privacy, the Prize has been criticized by privacy advocates. In 2007 two researchers from The University of Texas at Austin were able to identify individual users by matching the data sets with film ratings on the Internet Movie Database.[29][30]

On December 17, 2009, four Netflix users filed a class action lawsuit against Netflix, alleging that Netflix had violated U.S. fair trade laws and the Video Privacy Protection Act by releasing the datasets.[31] There was public debate about privacy for research participants. On March 19, 2010, Netflix reached a settlement with the plaintiffs, after which they voluntarily dismissed the lawsuit.

Pandora Radio

From Wikipedia, the free encyclopedia

Pandora Radio (also known as **Pandora Internet Radio** or simply **Pandora**) is a music streaming and automated music recommendation internet radio service powered by the Music Genome Project. The service, operated by Sirius XM Satellite Radio, is available in the United States.[4] The service plays songs that have similar musical traits.[5] The user then provides positive or negative feedback (as thumbs up or thumbs down) for songs chosen by the service, and the feedback is taken into account in the subsequent selection of other songs to play. The service can be accessed either through a web browser or with its mobile app. Pandora is a freemium service; basic features are free with advertisements or limitations, while additional features, such as improved streaming quality, music downloads and offline channels are offered via paid subscriptions.

In 2014, Pandora had about 76 million monthly users, and about a 70% share of the internet radio market in the U.S.[6]

Pandora's Promoted Stations rely on its core Music Genome Project. Overall, the Music Genome Project of more than 450 attributes assigned to each song with a human-curated database of recorded music.[7]

In February 2019, Sirius XM Satellite Radio acquired Pandora for $3.5 billion in stock.[8][9]

Pandora

Type of site	Subsidiary of Sirius XM Satellite Radio
Available in	English
Founded	January 2000; 19 years ago (as Savage Beast Technologies) Oakland, California, U.S.
Headquarters	Oakland, California, U.S
No. of locations	26
Area served	United States
Founder(s)	Will Glaser Jon Kraft Tim Westergren
Key people	Roger J. Lynch (CEO)[1]
Services	Internet radio
Employees	2,200+
Subsidiaries	Rdio (as of December 22, 2015), Next Big Sound

Contents [hide]

Recommender Systems: Pandora

- Pandora music recommendation
  - Recommend songs with similar scores
  - Recommend sequence of songs (playlists)

- Music Genome Project
  - Trained music analysts score each song based on hundreds of distinct musical characteristics.

Songza

From Wikipedia, the free encyclopedia

Songza was a free music streaming and recommendation service for Internet users in the United States and Canada.

Stating that its playlists are made by music experts, the service would recommend its users on various playlists based on time of day and mood or activity.[2][3] Songza offered playlists for activities such as waking up, working out, commuting, concentrating, unwinding, entertaining, and sleeping.[4] Users would vote songs up or down, and the service will adapt to the user's personal music preferences.[4] Users would find playlists not just based on artists, songs, or genres, but also based on themes, interests, and eras, such as "90s One-Hit Wonders", or "Music of Fashion Week".[5]

Songza was headquartered in the Long Island City neighborhood of the Queens borough of New York City, New York.[6]

On December 2, 2015, Google announced Songza would merge into Google Play Music on January 31, 2016.[7] As of February 1, 2016, the main site is offline, displaying a redirect to Google Play Music.

Songza

Type of site	Free internet radio
Available in	English
Headquarters	Long Island City, Queens, New York City, New York, United States
Owner	Alphabet Inc.
Created by	Aza Raskin and Scott Robbin
Website	songza.com
Alexa rank	▼ 7,598 (December 2015)[1]
Launched	November 8, 2007 (11 years ago)
Current status	Discontinued (now part of Google Play Music)

History [edit]

Amie Street acquired Songza, a product created by Aza Raskin and Scott Robbin, in October 2008.[8] In August 2010, Amie Street was sold to Amazon for an undisclosed amount.[9] Shortly after this the co-founders – CEO Elias Roman, COO Peter Asbill, CPO Elliott Breece and CCO Eric Davich – refocused their efforts on Songza.[2] The team discontinued the original version and relaunched a new alpha version of Songza, keeping nothing of the original product but the name.[10]

Over the next year the founders experimented with various iterations, when the app originally launched in 2010 "it was like a pre-Turntable.fm. A function called Social Radio allowed users to be DJs for their friends" stated PandoDaily.[3] This version of the app allowed it to be social and crowdsourced; the problem with it was that the service as it stood was not sufficiently differentiated from other services on the market and the quality of the crowd sourced playlists was low.[3] Following a year of testing various iterations of the alpha version of the app, Songza relaunched in beta on iPhone and Android apps on September 13, 2011, armed with a team of 25 expert music curators.[2][3][11][12]

Recommender Systems: Songza

Yahoo News Recommendations

- Recommendations of new articles for Today box on Yahoo's home page
  - 9,000 recommendations per minute
  - Sophisticated personalization algorithm
  - Based on demographic user attributes, the places they've visited when they've come to Yahoo in the past, and the stories they've already seen during that particular visit.
  - Team of editors prepare 50-100 news packages, algorithm ranks packages for user.
  - Has increased the click through rate by 270% since 2009.
  - Has helped editors to get better understanding of the interests of different user segments.

YAHOO! NEWS

Should the U.S. expand the Supreme Court?

Commentators say a knockdown battle over every open SCOTUS seat is demoralizing for the country and cements the perception of the court as a political body.

The Ideas Election »

1,254 people reacting

Yahoo News Exclusives

Michael Isikoff

Ocasio-Cortez backs boycott of New York Post over cover attacking Ilhan Omar

Matt Bai

Stephen Miller stokes Trump's nationalist vision

What you won't see in the Mueller report Thursday

Buttigieg shuts down antigay hecklers at rally

Cher: Trump is an 'ignorant thug with a lizard brain'

Red or blue states more dependent on government?

College student falls to her death off 'treacherous' cliff

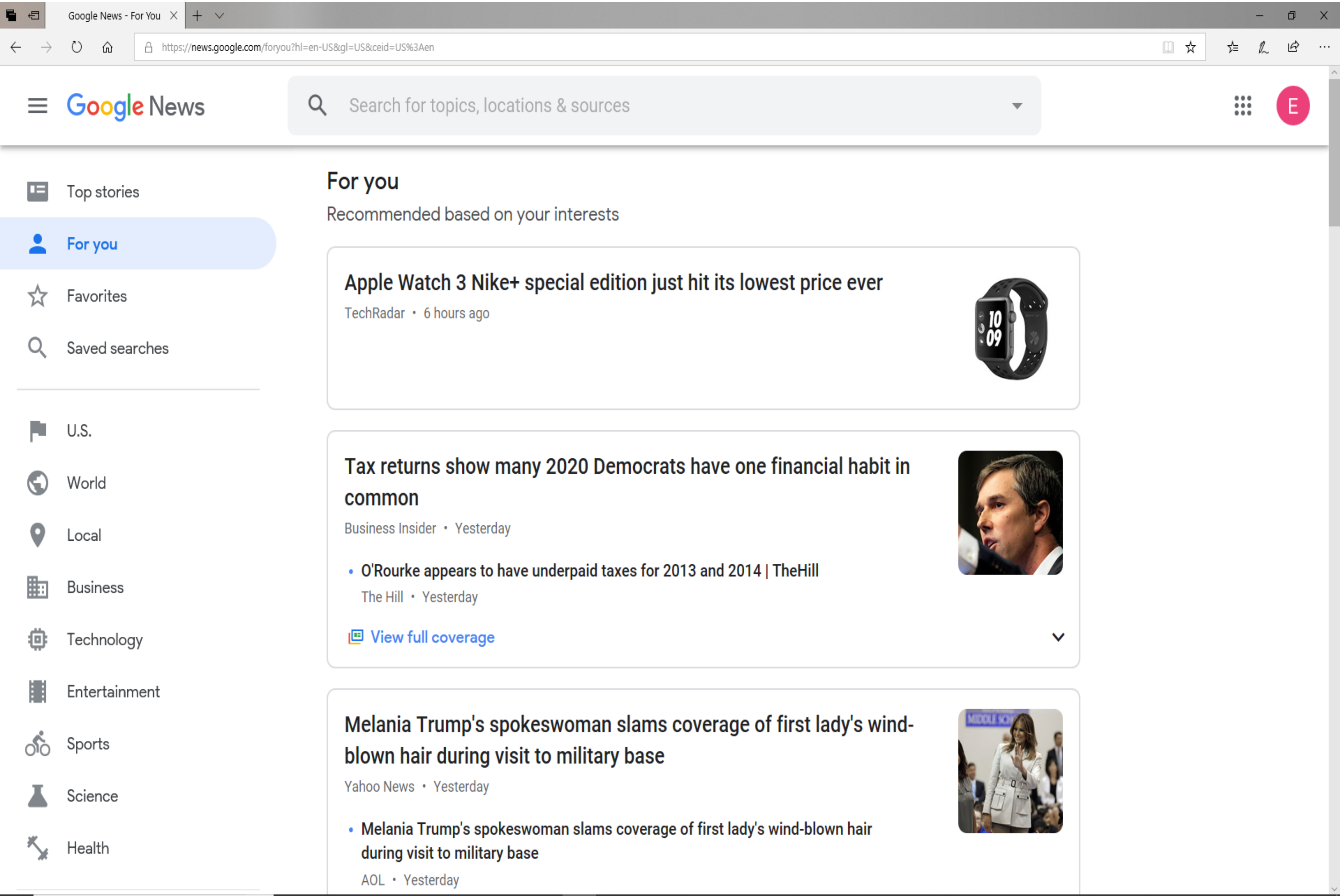
Politics Reuters

285

Google News Portal

- Aggregates news articles from several thousand sources and Displays them to signed-in users in a personalized way
  - **Collaborative** recommendation approach based on
    - the click history of the active user and
    - the history of the larger community
  - Main challenges
    - Vast number of articles and users
    - Generate recommendation list in real time (**< 1 second**)
    - Constant stream of new items
    - Immediately react to user interaction
  - Significant efforts with respect to algorithms, engineering, and parallelization are required

Google News Personalization Engine



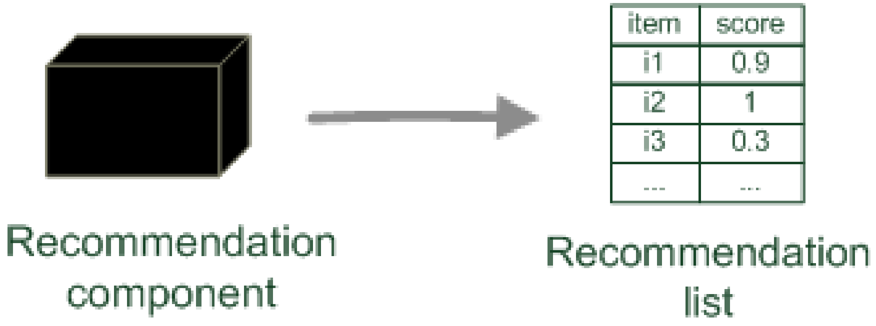
Recommendation Tasks

- Rating prediction: Predict the rating of target user for target item, e.g. predict Joe's rating for Titanic.
- Top-N recommendation: Predict the top-N highest-rated items among the items not yet rated by target user.
- Link recommendation (only in social networks): Predict the top-N users to which the target user is most likely to connect.

- Seen as a function:
  - Given:
    - User model (e.g. ratings, preferences, demographics, situational context)
    - Items (with or without description of item characteristics)
  - Find:
    - **Relevance** score. Used for ranking.

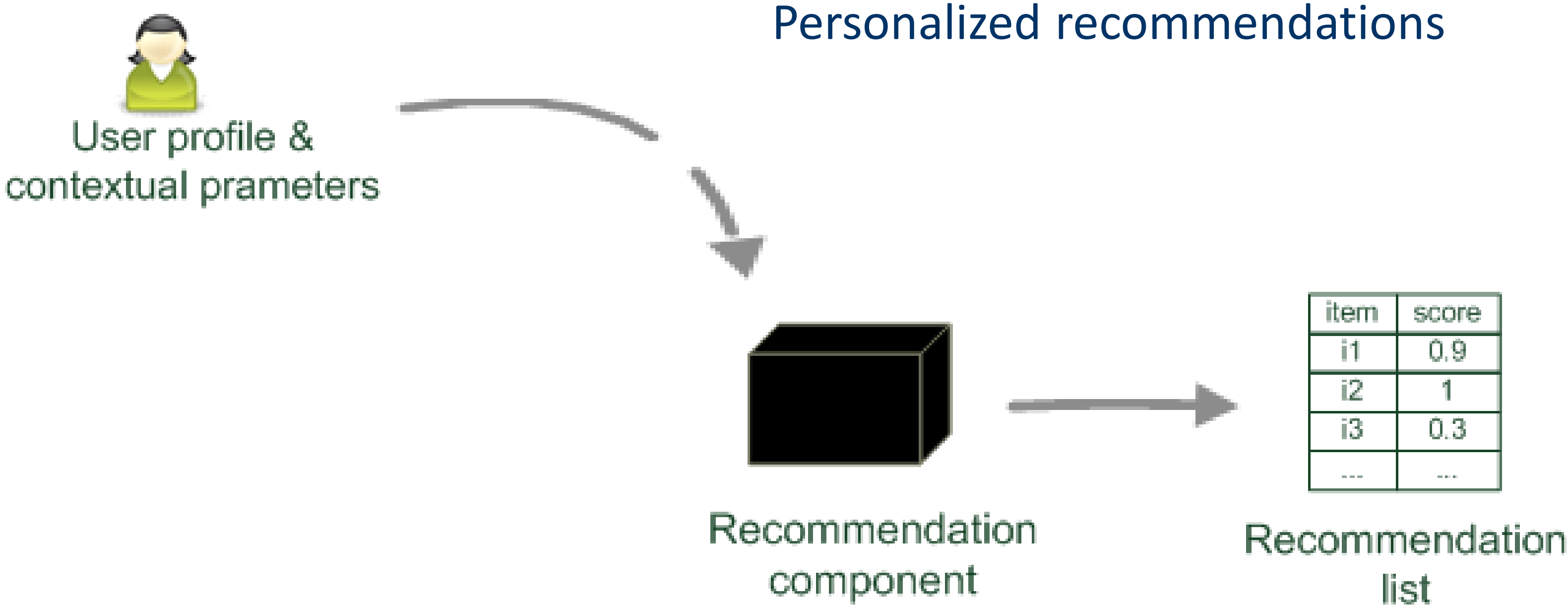
Paradigms of Recommender Systems

Recommender systems reduce information overload by estimating relevance



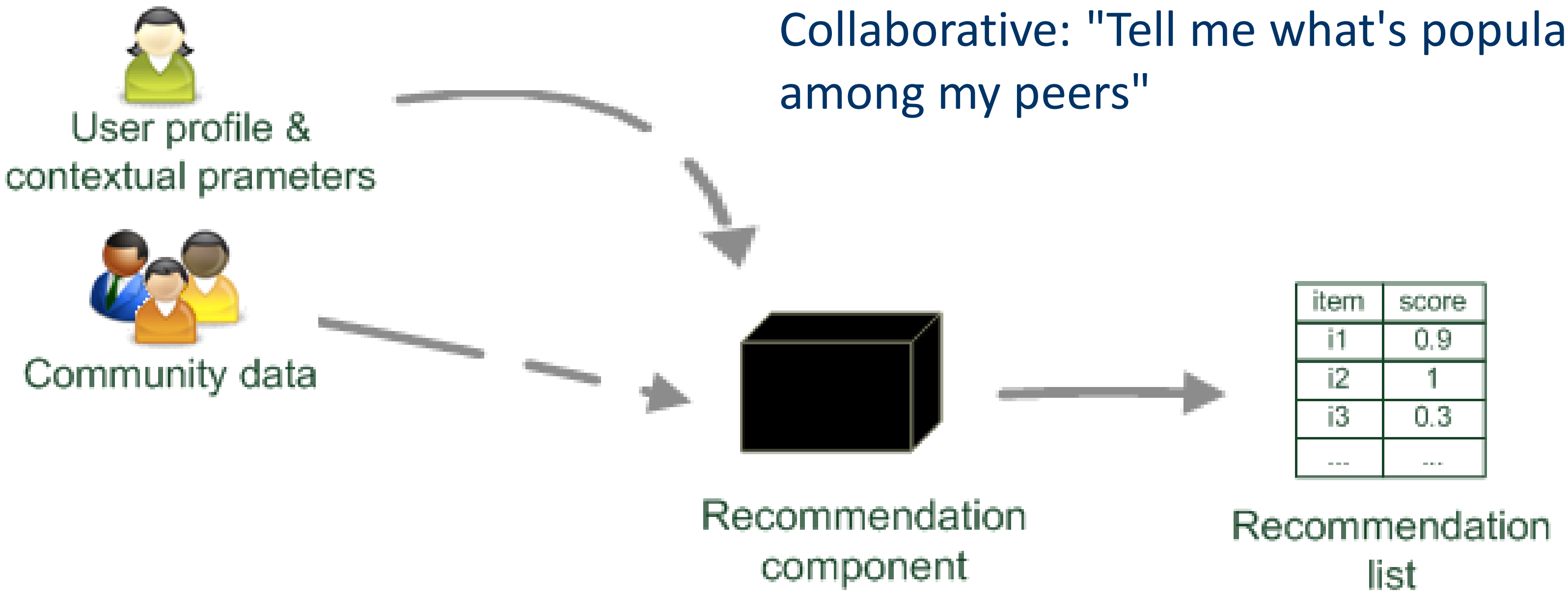
Paradigms of Recommender Systems

Personalized recommendations

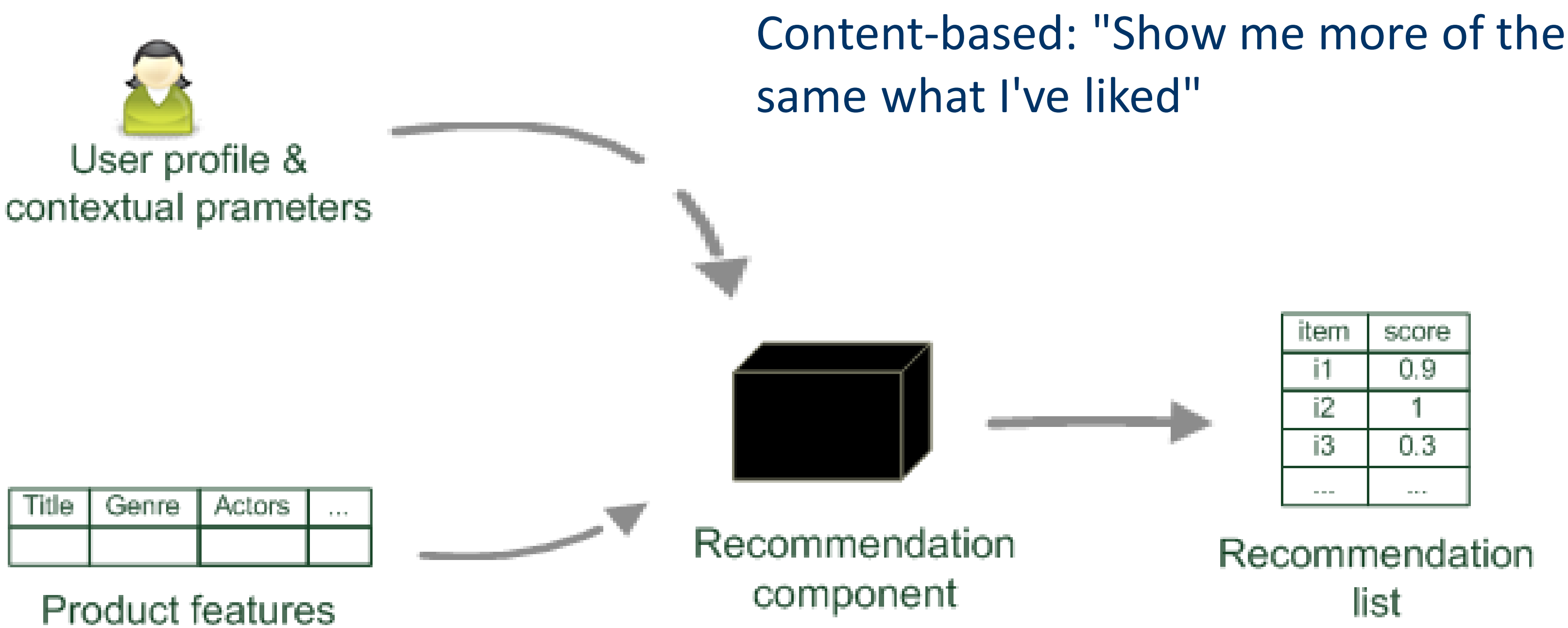


Paradigms of Recommender Systems

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

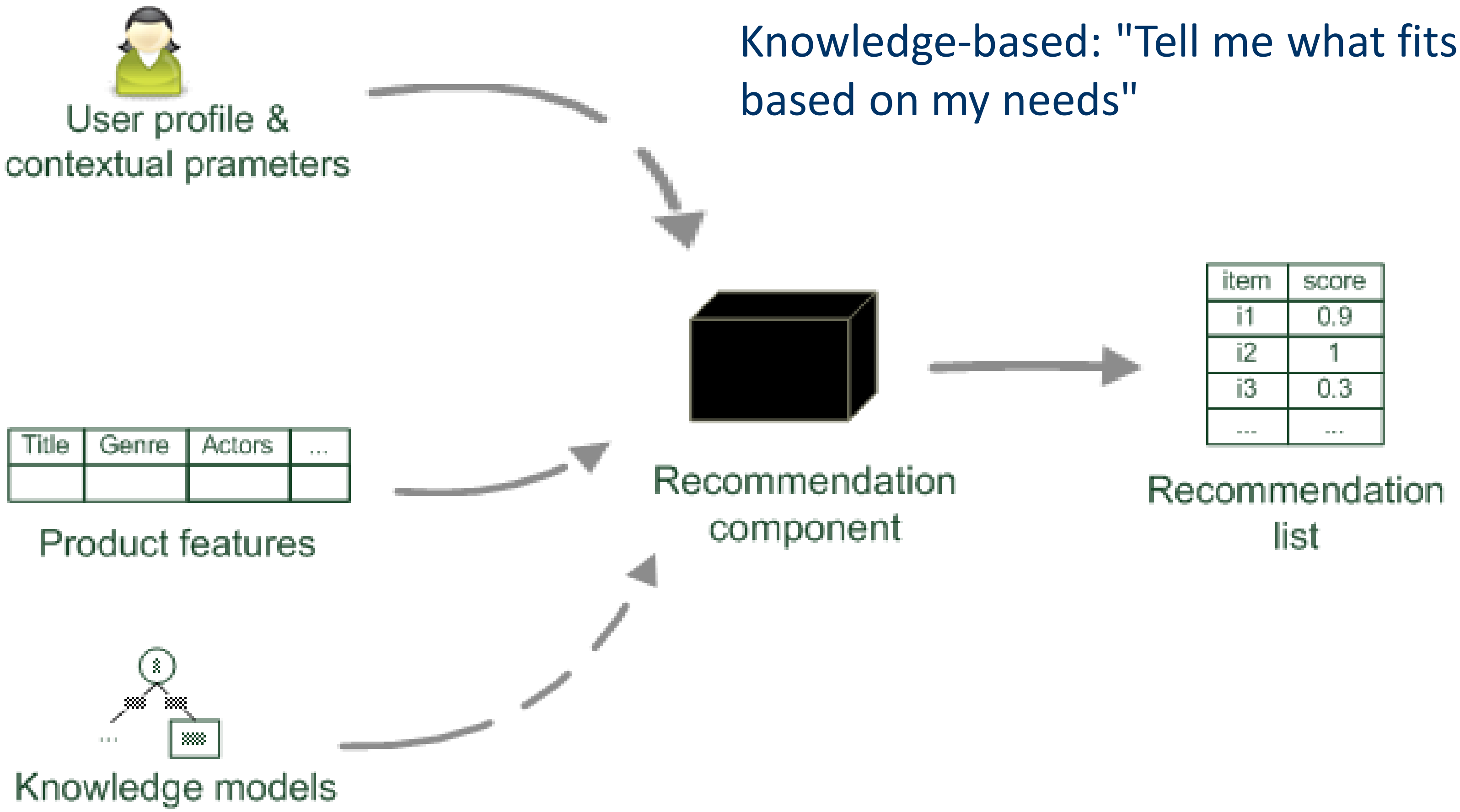


Paradigms of Recommender Systems



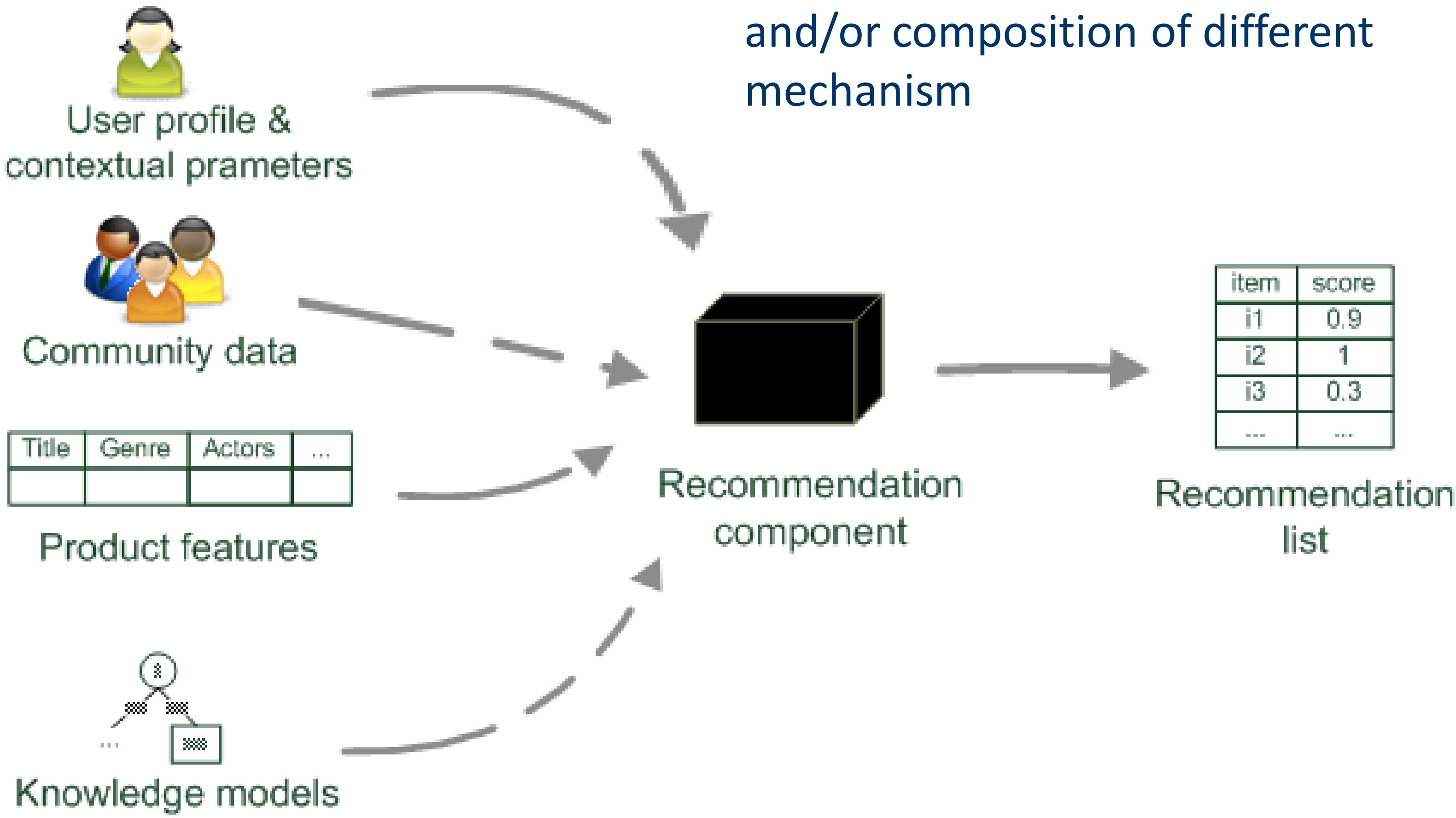
Paradigms of Recommender Systems

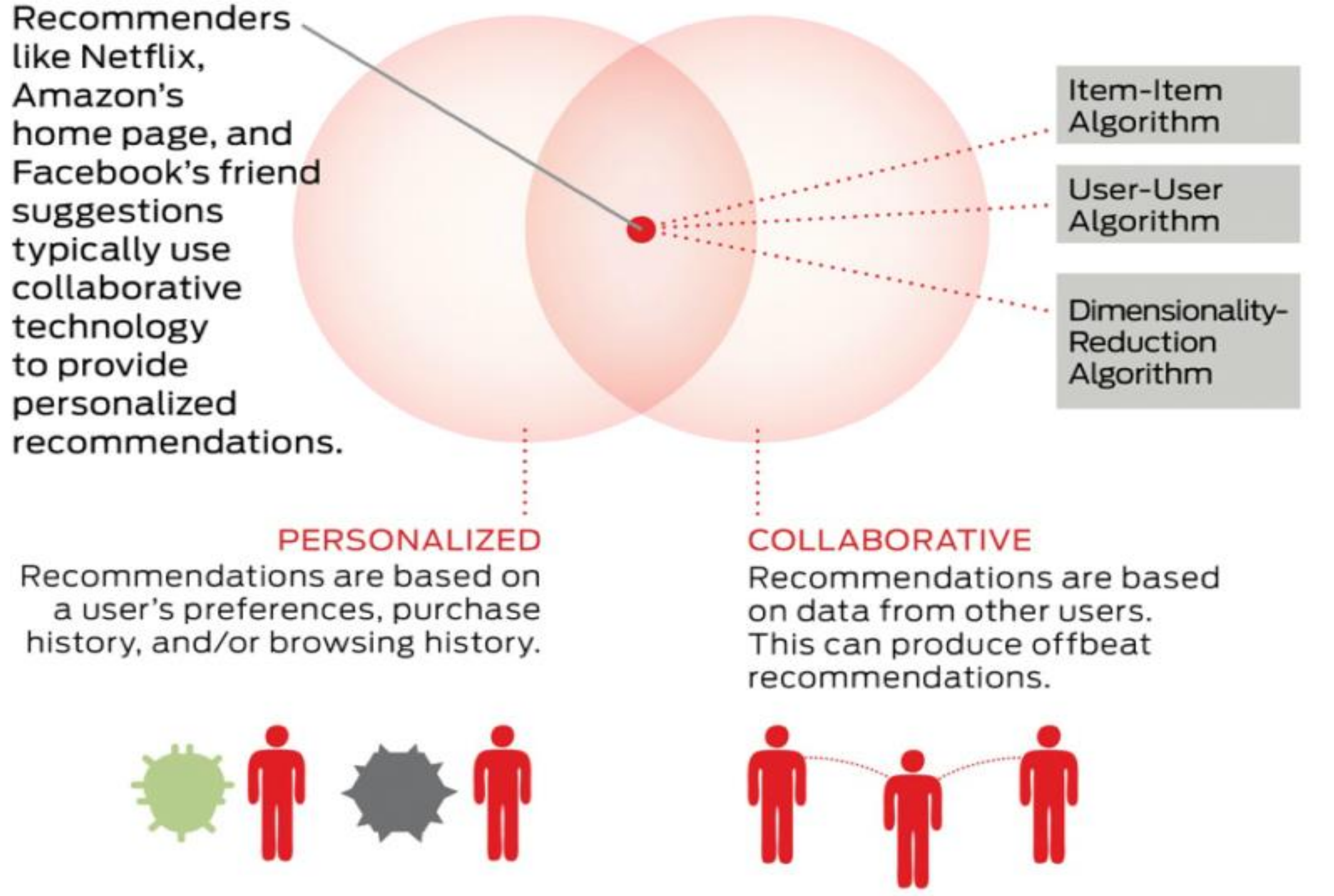
Knowledge-based: "Tell me what fits based on my needs"

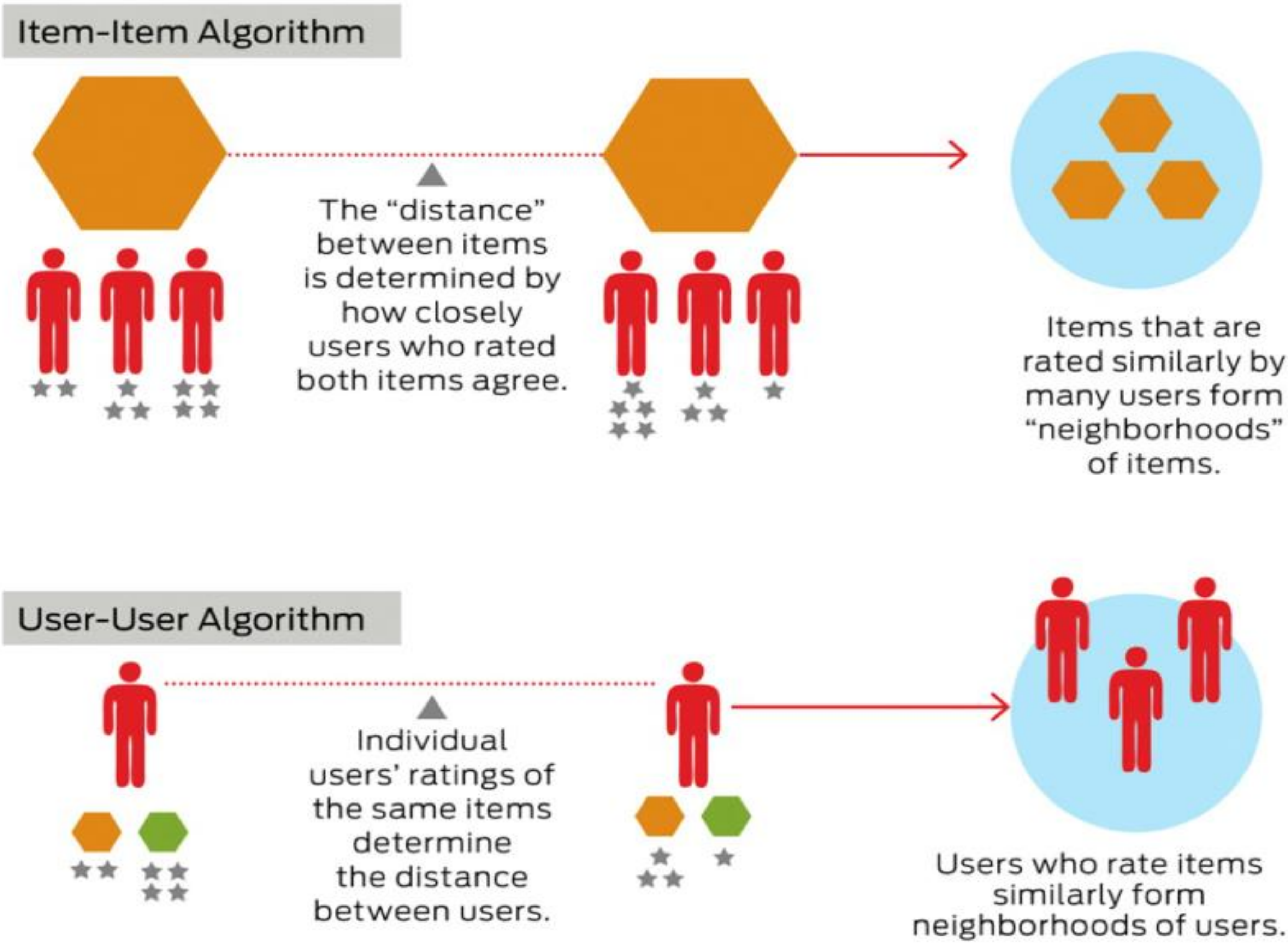


Paradigms of Recommender Systems

Hybrid: combinations of various inputs and/or composition of different mechanism

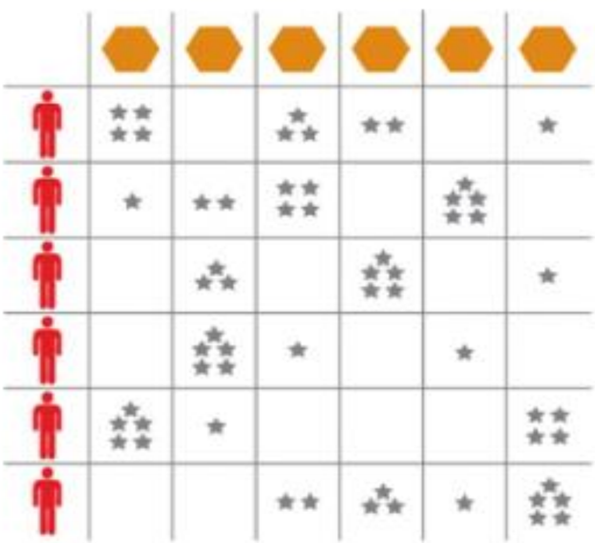



Recommenders like Netflix, Amazon's home page, and Facebook's friend suggestions typically use collaborative technology to provide personalized recommendations.
Item-Item Algorithm
User-User Algorithm
Dimensionality-Reduction Algorithm
PERSONALIZED
Recommendations are based on a user's preferences, purchase history, and/or browsing history.
COLLABORATIVE
Recommendations are based on data from other users. This can produce offbeat recommendations.


Item-Item Algorithm
The "distance" between items is determined by how closely users who rated both items agree.
Items that are rated similarly by many users form "neighborhoods" of items.
User-User Algorithm
Individual users' ratings of the same items determine the distance between users.
Users who rate items similarly form neighborhoods of users.

Dimensionality-Reduction Algorithm (part 1)

A dimensionality-reduction algorithm starts with a customer product-preference matrix.



The algorithm then compresses and factors the matrix, identifying important taste dimensions.

.8	.2	.1	.5
0	.7	.8	.3
.8	.6	.4	0
.5	.8	.2	.7
0	.3	.7	.1
.2	.1	.3	.2

.2	.7	0	.3	.4	.7
.8	.6	.2	.8	.6	.5
.1	.7	.5	.4	.2	0
.3	.4	.6	.8	.3	.6

Dimensionality-Reduction Algorithm (part 2)

The resulting taste signature pretty accurately—if abstractly—represents user tastes.

	.8	.2	.1	.5
	0	.7	.8	.3
	.8	.6	.4	0
	.5	.8	.2	.7
	0	.3	.7	.1
	.2	.1	.3	.2

	.2	.7	0	.3	.4	.7
	.8	.6	.2	.8	.6	.5
	.1	.7	.5	.4	.2	0
	.3	.4	.6	.8	.3	.6

User profiles come from many sources.

User Data

Your data comes from items you rate, enlarge, or look at multiple times; items you place on wish lists; and items you actually purchase.

Recommenders store nearly every action you take when logged into a site, which raises privacy concerns.

Navigation

Recommenders follow your path through a site.

The recommender system builds a profile of long-term preferences.

Recommenders can use these different sources of preference information together or separately.

Collaborative Filtering (CF)

- Is the most prominent approach to generate recommendations
  - Used by large, commercial e-commerce sites
  - Well-understood, various algorithms and variations exist
  - Applicable in many domains (book, movies, DVDs, ..)
- Approach
  - Use the "**wisdom of the crowd**" to recommend items (collective intelligence)

- Basic assumptions:
  - Users give ratings to items (implicitly or explicitly)
  - Customers who had similar tastes in the past will have similar tastes in the future

Pure CF Approaches

- Input
  - Only a matrix of given user–item ratings

- Output types
  - A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
  - A top-N list of recommended items

CF: User-based Nearest-Neighbor

- **The basic technique**
  - Given an "active user" (Alice) and an item i not yet seen by Alice
    - find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item i
    - use, e.g. the average of their ratings to predict, if Alice will like item i
    - do this for all items Alice has not seen and recommend the best-rated
- **Basic assumption and idea**
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

CF: User-based Nearest-Neighbor

- Example
  - A database of ratings of the current user, Alice, and some other users is given:

	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

  - Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen

CF: User-based Nearest-Neighbor

- First questions:
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	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

Measuring User Similarity

- **A popular similarity measure in user-based CF: Pearson correlation**

  a, b : users

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

  P : set of items, rated both by a and b

  – Possible similarity values between -1 and 1

$$\boldsymbol{sim(a,b)} = \frac{\sum_{\boldsymbol{p} \in \boldsymbol{P}}(\boldsymbol{r_{a,p}} - \bar{\boldsymbol{r}}_{\boldsymbol{a}})(\boldsymbol{r_{b,p}} - \bar{\boldsymbol{r}}_{\boldsymbol{b}})}{\sqrt{\sum_{\boldsymbol{p} \in \boldsymbol{P}}\left(\boldsymbol{r_{a,p}} - \bar{\boldsymbol{r}}_{\boldsymbol{a}}\right)^{2}} \sqrt{\sum_{\boldsymbol{p} \in \boldsymbol{P}}\left(\boldsymbol{r_{b,p}} - \bar{\boldsymbol{r}}_{\boldsymbol{b}}\right)^{2}}}$$

Measuring User Similarity

- **A popular similarity measure in user-based CF: Pearson correlation**

  a, b : users

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

  P : set of items, rated both by a and b

  – Possible similarity values between -1 and 1

	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

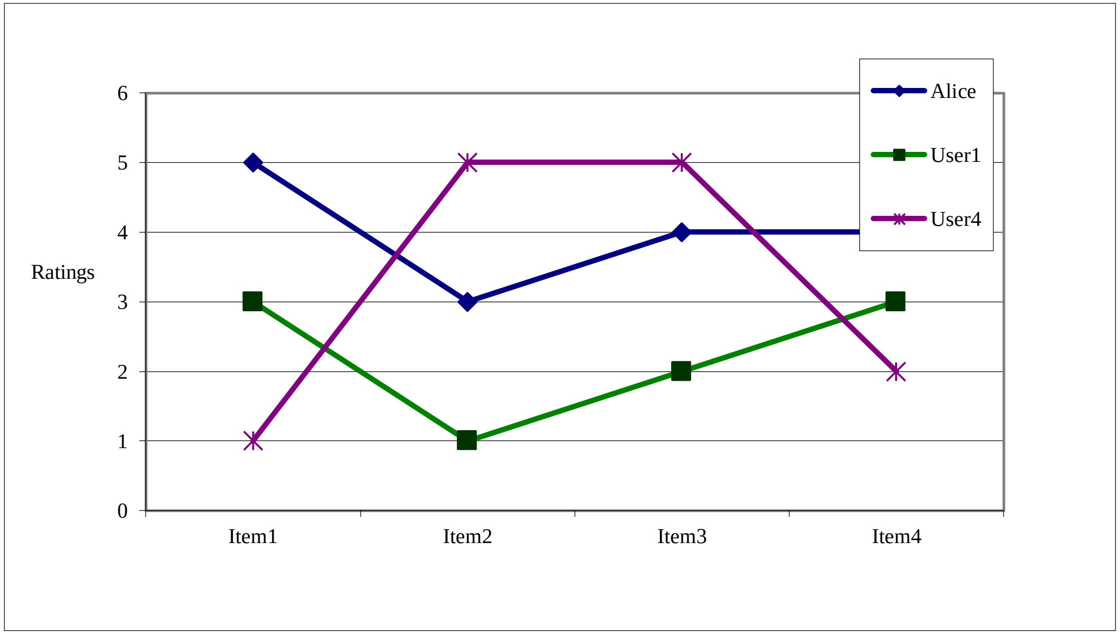
sim = 0.70

sim = 0.00

sim = -0.79

Pearson Correlation

❖ Takes differences in rating behavior into account



❖ Works well in usual domains, compared with alternative measures such as cosine similarity

Making Predictions

- A common prediction function:

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

- 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 a as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Item-based Collaborative Filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	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

The cosine Similarity Measure

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

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

- **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

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

Making Predictions

- A common prediction function:

$$\boldsymbol{pred(u,p)} = \frac{\sum_{i \in ratedItem(u)} \boldsymbol{sim(i,p) * r_{u,i}}}{\sum_{i \in ratedItem(u)} \boldsymbol{sim(i,p)}}$$

  - Neighborhood size is typically also limited to a specific size
  - Not all neighbors are taken into account for the prediction
  - An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable" (Herlocker et al. 2002)

Pre-processing

- Item-based filtering does not solve the scalability problem itself
- Pre-processing approach by Amazon.com (in 2003)
  - Calculate all pair-wise item similarities in advance
  - The neighborhood to be used at run-time is typically rather small, because only the items which the user has rated are taken into account
  - Item similarities are supposed to be more stable than user similarities
- Memory requirements
  - Up to N^2 pair-wise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible
    - Minimum threshold for co-ratings
    - Limit the neighborhood size (might affect recommendation accuracy)

More on Ratings – Explicit Ratings

- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- Research topics
  - Optimal granularity of scale; indication that 10-point scale is better accepted in the movie domain.
  - An even more fine-grained scale was chosen in the joke recommender discussed by Goldberg et al. (2001), where a continuous scale (from −10 to +10) and a graphical input bar were used
    - No precision loss from the discretization
    - User preferences can be captured at a finer granularity
    - Users actually "like" the graphical interaction method

  - Multidimensional ratings (multiple ratings per movie such as ratings for actors and sound)
- Main problems
  - number of available ratings could be too small → sparse rating matrices → poor recommendation quality
  - How to stimulate users to rate more items?

More on Ratings – Implicit Ratings

- Typically collected by the web shop or application in which the recommender system is embedded
- When a customer buys an item, for instance, many recommender systems interpret this behavior as a positive rating
- Clicks, page views, time spent on some page, demo downloads …
- Implicit ratings can be collected constantly and do not require additional efforts from the side of the user
- Main problem
  - One cannot be sure whether the user behavior is correctly interpreted
  - For example, a user might not like all the books he or she has bought; the user also might have bought a book for someone else
- Implicit ratings can be used in addition to explicit ones

Data Sparsity Problems

- Cold start problem
  - How to recommend new items? What to recommend to new users?
- Straightforward approaches
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
  - Assign default values
- Alternatives
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be too small to make good predictions
    - Assume "**transitivity**" of neighborhoods

Examples for Sparse Datasets

- **Recursive CF** (Zhang and Pu 2007)
  - Assume there is a very close neighbor n of u who however has not rated the target item i yet.
  - Idea:
    - Apply CF-method recursively and predict a rating for item i for the neighbor
    - Use this predicted rating instead of the rating of a more distant direct neighbor

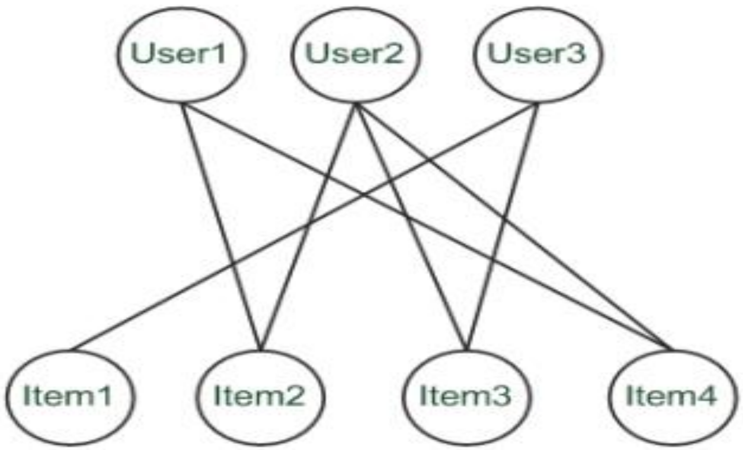
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	?
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0.85

Predict rating for User1

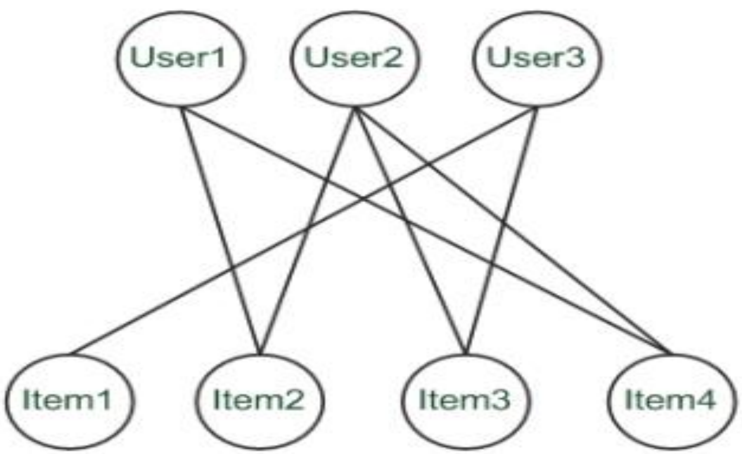
Graph-based Methods

- "Spreading activation" (Huang et al. 2004)
  - Exploit the supposed "**transitivity**" of customer tastes and thereby augment the matrix with additional information
  - Assume that we are looking for a recommendation for ***User1***
  - When using a standard CF approach, ***User2*** will be considered a peer for ***User1*** because they both bought *Item2* and *Item4*
  - Thus *Item3* will be recommended to *User1* because the nearest neighbor, *User2,* also bought or liked it



Graph-based Methods

- "Spreading activation" (Huang et al. 2004)
  - In a standard user-based or item-based CF approach, paths of length 3 will be considered – that is, *Item3* is relevant for ***User1*** because there exists a three-step path (*User1–Item2–User2–Item3*) between them
  - Because the number of such paths of length 3 is small in sparse rating databases, the idea is to also consider longer paths (indirect associations) to compute recommendations
  - Using path of length 5, for instance



More Approaches

- Plethora of different techniques proposed in the last years:
  - Matrix factorization techniques (Dimensionality Reduction)
    - singular value decomposition (SVD), principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches

2000: *Application of Dimensionality Reduction in Recommender System*, B. Sarwar et al., WebKDD Workshop

- Basic idea: Trade more complex offline model building for faster online prediction generation
- Singular Value Decomposition for dimensionality reduction of rating matrices
  - Captures important factors/aspects and their weights in the data
  - factors can be genre, actors but also non-understandable ones
  - Assumption that *k* dimensions capture the signals and filter out noise (*k* = 20 to 100)
- Constant time to make recommendations

Matrix Factorization

- Informally, the SVD theorem (Golub and Kahan 1965) states that a given matrix M can be decomposed into a product of three matrices as follows

$$M = U \times \Sigma \times V^T$$

  - where U and V are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*
- We can approximate the full matrix by observing only the most important features – those with the largest singular values
- In the example, we calculate U, V, and Σ (with the help of some linear algebra software) but retain only the two most important features by taking only the first two columns of U and V^T

Example for SVD

- SVD: $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

= 3 + 0.84 = 3.84

Association Rule Mining

- **Commonly used for shopping behavior analysis**
  - aims at detection of rules such as
    "If a customer purchases beer then he also buys diapers in 70% of the cases"
- **Association rule mining algorithms**
  - can detect rules of the form X → Y (e.g., beer → diapers) from a set of sales transactions $D = \{t_1, t_2, \dots t_n\}$
  - measure of quality: support, confidence
    - used e.g. as a threshold to cut off unimportant rules
  - let $\sigma(X) = \frac{|\{x \mid x \subseteq ti, ti \in D\}|}{|D|}$
  - support = $\frac{\sigma(X \cup Y)}{|D|}$, confidence = $\frac{\sigma(X \cup Y)}{\sigma(X)}$

Association Rule Mining

	Item1	Item2	Item3	Item4	Item5
Alice	1	0	0	0	?
User1	1	0	1	0	1
User2	1	0	1	0	1
User3	0	0	0	1	1
User4	0	1	1	0	0

- Simplest approach
  - transform 5-point ratings into binary ratings (1 = above user average)
- Mine rules such as
  - Make recommendations for Alice
    - Item1 → Item5: support (2/4), confidence (2/2) (without Alice)
- Make recommendations for Alice
  - Determine "relevant" rules based on Alice's transactions (the above rule will be relevant as Alice bought Item1)
  - Determine items not already bought by Alice
  - Sort the items based on the rules' confidence values

Probabilistic Methods

- **Basic idea (simplistic version for illustration):**
  - given the user/item rating matrix
  - determine the probability that user Alice will like an item i
  - base the recommendation on such these probabilities
- **Calculation of rating probabilities based on Bayes Theorem**
  - How probable is rating value "1" for Item5 given Alice's previous ratings?
  - Corresponds to conditional probability P(Item5=1 | X), where
    - X = Alice's previous ratings = (Item1 =1, Item2=3, Item3= ...)
  - Can be estimated based on Bayes' Theorem

$$\boldsymbol{P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}} \qquad \boldsymbol{P(Y|X) = \frac{\prod_{i=1}^{d} P(X_i|Y) \times P(Y)}{P(X)}}$$

  - Assumption: Ratings are independent (?)

Calculations

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

5-Class Classification

X = (Item1 =1, Item2=3, Item3= …)

$$
\begin{aligned}
&P(X|Item5=1) \\
&\quad = P(Item1=1|Item5=1) \times P(Item2=3|Item5=1) \\
&\quad \times P(Item3=3|Item5=1) \times P(Item4=2|Item5=1) \\
&\quad = \frac{2}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \approx 0.125
\end{aligned}
$$

$$
\begin{aligned}
&P(X|Item5=2) \\
&\quad = P(Item1=1|Item5=2) \times P(Item2=3|Item5=2) \\
&\quad \times P(Item3=3|Item5=2) \times P(Item4=2|Item5=2) \\
&\quad = \frac{0}{0} \times \cdots \times \cdots \times \cdots = 0
\end{aligned}
$$

Collaborative Filtering Issues

- Pros:
  - well-understood, works well in some domains, no knowledge engineering required
- Cons:
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results
- What is the best CF method?
  - In which situation and which domain? Inconsistent findings; always the same domains and data sets; differences between methods are often very small (1/100)
- How to evaluate the prediction quality?
  - Commonly used: MAE/RMSE, but what does an MAE of 0.7 actually mean?

MAE/RMSE

- **Mean Absolute Error (*MAE*).** The average absolute deviation between a predicted rating (p) and the user's true rating (r)

$$MAE = \frac{\sum_{ij} |\hat{r}_{ij} - r_{ij}|}{n}$$

  - $NMAE = MAE/(r_{max} - r_{min})$

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

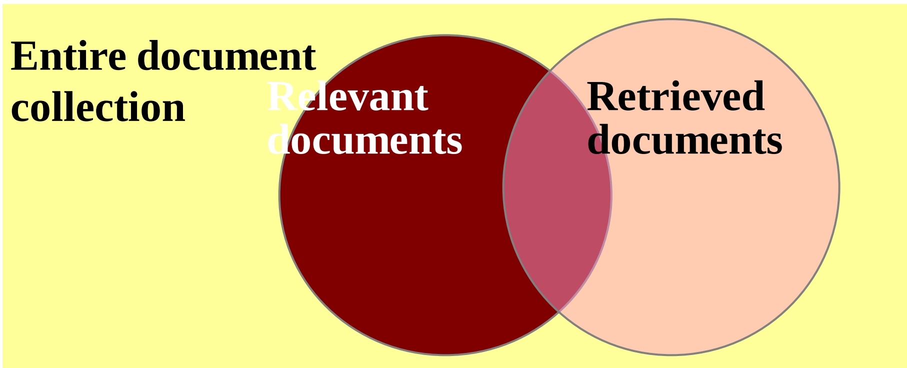
$$RMSE = \sqrt{\frac{1}{n}\sum_{i,j}(\hat{r}_{ij} - r_{ij})^2}$$

MAE/RMSE

Item	*Predicted Rating*	*True Rating*
1	*1*	*3*
2	*2*	*5*
3	*3*	*3*
4	*4*	*2*
5	*4*	*1*

$$
\begin{aligned}
MAE &= \frac{|1-3|+|2-5|+|3-3|+|4-2|+|4-1|}{5} = 2 \\
NMAE &= \frac{MAE}{5-1} = 0.5 \\
RMSE &= \sqrt{\frac{(1-3)^2+(2-5)^2+(3-3)^2+(4-2)^2+(4-1)^2}{5}} \\
&= 2.28
\end{aligned}
$$

Precision and Recall



	retrieved	not retrieved
relevant	retrieved & relevant	Not retrieved & relevant
irrelevant	retrieved & irrelevant	not retrieved but irrelevant

$$recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents}$$

$$precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ documents\ retrieved}$$

To consolidate Precision & Recall into 1 measure: **F = 2PR/(P+R)**

The YouTube Video Recommendation System

James Davidson
Google Inc
davidson@google.com

Benjamin Liebald
Google Inc
liebald@google.com

Junning Liu
Google Inc
ljn@google.com

Palash Nandy
Google Inc
palash@google.com

Taylor Van Vleet
Google Inc
tvv@google.com

ABSTRACT

We discuss the video recommendation system in use at YouTube, the world's most popular online video community. The system recommends personalized sets of videos to users based on their activity on the site. We discuss some of the unique challenges that the system faces and how we address them. In addition, we provide details on the experimentation and evaluation framework used to test and tune new algorithms. We also present some of the findings from these experiments.

Categories and Subject Descriptors

H.3 [Information Systems]: Information Storage and Retrieval; H.4 [Information Systems]: Information Systems

and every minute, users upload more than 24 hours of video to YouTube.

In this paper, we present our video recommendation system, which delivers personalized sets of videos to signed in users based on their previous activity on the YouTube site (while recommendations are also available in a limited form to signed out users, we focus on signed in users for the remainder of this paper). Recommendations are featured in two primary locations: The YouTube home page (`http://www.youtube.com`) and the "Browse" page at `http://www.youtube.com/videos`. An example of how recommendations are presented on the homepage can be found in Figure 1.

1.1 Goals

days requiring constant freshness of recommendation.

2. SYSTEM DESIGN

The overall design of the recommendation system is guided by the goals and challenges outlined above: We want recommendations to be reasonably recent and fresh, as well as diverse and relevant to the user's recent actions. In addition, it's important that users understand why a video was recommended to them.

The set of recommended videos videos is generated by using a user's personal activity (watched, favorited, liked videos) as seeds and expanding the set of videos by traversing a co-visitation based graph of videos. The set of videos is then ranked using a variety of signals for relevance and diversity.

From an engineering perspective, we want individual components of the system to be decoupled from each other, allowing them to be understood and debugged in isolation. Given that our system is part of the larger YouTube ecosystem, recommendations also needs to be resilient to failure and degrade gracefully in case of partial failures. As a consequence, we strive to minimize complexity in the overall system.

2.1 Input data

During the generation of personalized video recommendations we consider a number of data sources. In general, there are two broad classes of data to consider: 1) content data, such as the raw video streams and video metadata such as title, description, etc, and 2) user activity data, which can further be divided into explicit and implicit categories. Explicit activities include rating a video, favoriting/liking a video, or subscribing to an uploader. Implicit activities are datum generated as a result of users watching and interact-

2.2 Related Videos

One of the building blocks of the recommendation system is the construction of a mapping from a video v_i to a set of similar or *related* videos R_i. In this context, we define similar videos as those that a user is likely to watch after having watched the given *seed video* v. In order to compute the mapping we make use of a well-known technique known as association rule mining [1] or co-visitation counts. Consider sessions of user watch activities on the site. For a given time period (usually 24 hours), we count for each pair of videos (v_i, v_j) how often they were co-watched within sessions. Denoting this co-visitation count by c_{ij}, we define the *relatedness score* of video v_j to base video v_i as:

$$r(v_i, v_j) = \frac{c_{ij}}{f(v_i, v_j)} \tag{1}$$

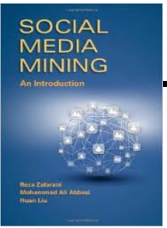
where c_i and c_j are the total occurrence counts across all sessions for videos v_i and v_j, respectively. $f(v_i, v_j)$ is a normalization function that takes the "global popularity" of both the seed video and the candidate video into account. One of the simplest normalization functions is to simply divide by the product of the videos' global popularity: $f(v_i, v_j) = c_i \cdot c_j$. Other normalization functions are possible. See [6] for an overview of possible choices. When using the simple product of cardinalities for normalization, c_i is the same for all candidate related videos and can be ignored in our setting, so we are normalizing only by the candidate's global popularity. This essentially favors less popular videos over popular ones.

We then pick the set of related videos R_i for a given seed video v_i as the top N candidate videos ranked by their scores $r(v_i, v_j)$. Note that in addition to only picking the top N videos, we also impose a minimum score threshold. Hence,

Recommendation to Groups

- Find content of interest to all members of a group of socially acquainted individuals

- Examples:
  - A movie for friends to watch together
  - A travel destination for a family to spend a break
  - A good restaurant for colleagues to have lunch
  - A music to be played in a public area



Recommendation to Groups

Maximizing Average Satisfaction

$$R_i = \frac{1}{n}\sum_{u \in G} r_{u,i}$$

- ❖Average everyone's ratings and choose the max

Least Misery

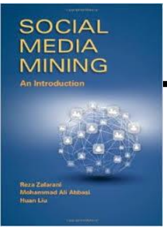
$$R_i = \min_{u \in G} r_{u,i}$$

- ❖This approach tries to minimize the dissatisfaction among group's members (max of all mins)

Most Pleasure

$$R_i = \max_{u \in G} r_{u,i}$$

- ❖The maximum of individuals' maximum ratings is taken as group's rating



Recommendation to Groups

Table 9.3: User-Item Matrix

	Soda	Water	Tea	Coffee
John	1	3	1	1
Joe	4	3	1	2
Jill	2	2	4	2
Jorge	1	1	3	5
Juan	3	3	4	5

Assuming John, Jill and Juan are traveling together this time.

Average Satisfaction

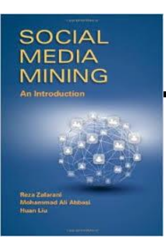
$$\begin{aligned} R_{Soda} &= \frac{1+2+3}{3} = 2. \\ R_{Water} &= \frac{3+2+3}{3} = 2.66 \\ R_{Tea} &= \frac{1+4+4}{3} = 3. \\ R_{Coffee} &= \frac{1+2+5}{3} = 2.66 \end{aligned}$$

Least Misery

$$\begin{aligned} R_{Soda} &= \min\{1,2,3\} = 1 \\ R_{Water} &= \min\{3,2,3\} = 2 \\ R_{Tea} &= \min\{1,4,4\} = 1 \\ R_{Coffee} &= \min\{1,2,5\} = 1 \end{aligned}$$

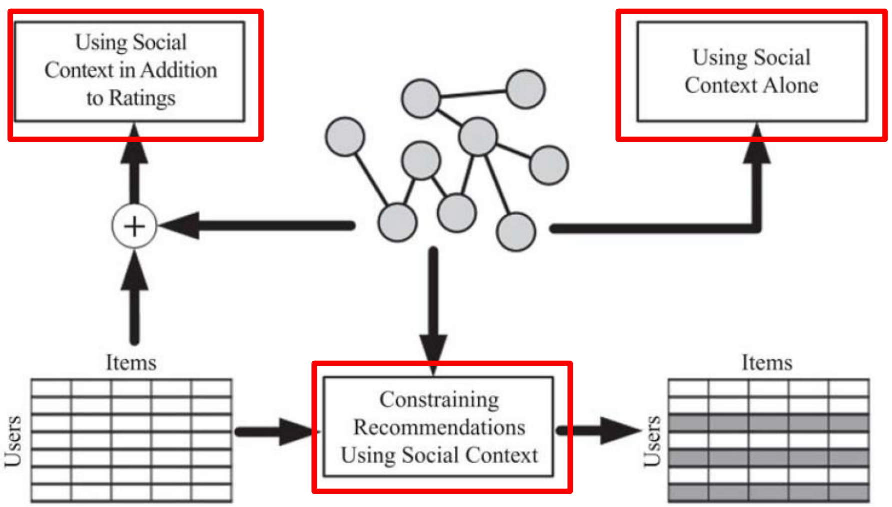
Most Pleasure

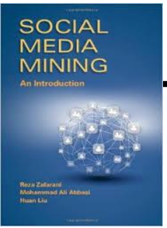
$$\begin{aligned} R_{Soda} &= \max\{1,2,3\} = 3 \\ R_{Water} &= \max\{3,2,3\} = 3 \\ R_{Tea} &= \max\{1,4,4\} = 4 \\ R_{Coffee} &= \max\{1,2,5\} = 5 \end{aligned}$$



Recommendation Using Social Context

- In social media, in addition to ratings of products, there is additional information
  - E.g., the friendship network
- This information can be used to improve recommendations
  - Assuming that friends have an impact on the ratings ascribed by the individual.
  - This impact can be due to **homophily**, **influence**, or **confounding**



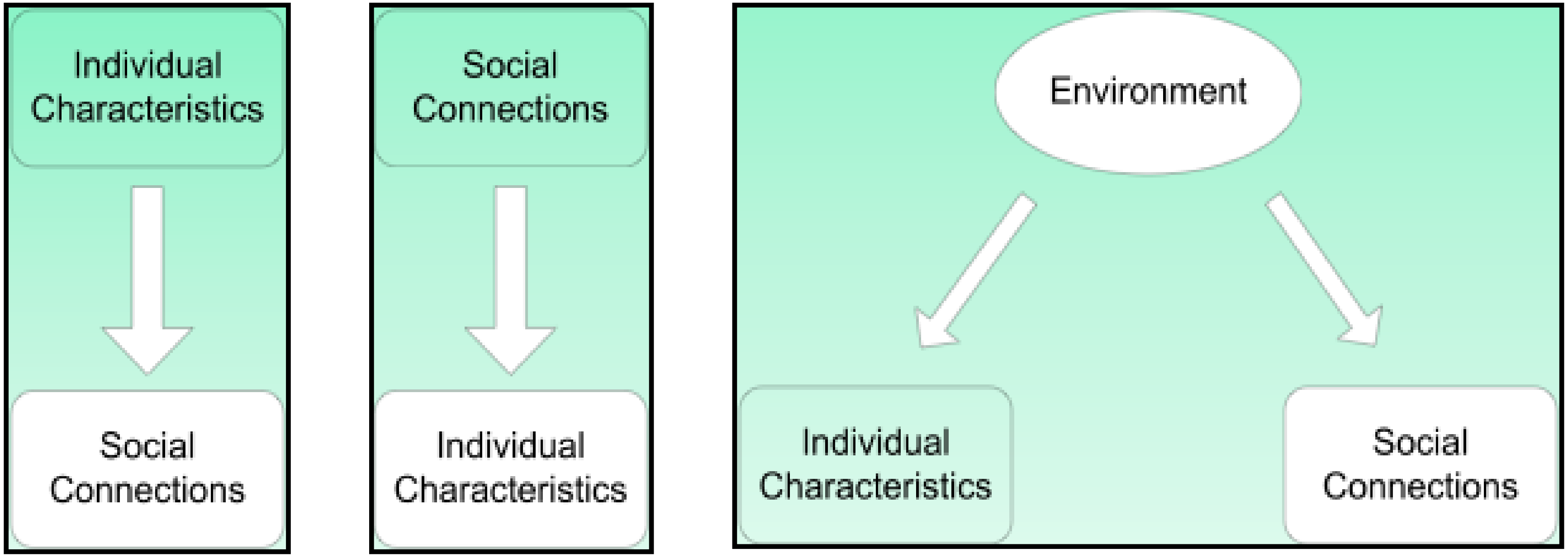


Correlation in Social Networks

Three major social processes to explain correlation between behaviors/attributes of <u>adjacent nodes</u> in a social network are:

- *Homophily*, *confounding*, *and influence*

homophily | influence | Confounding

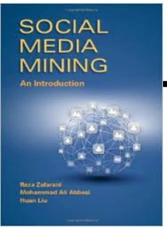


Recommendation Using Social Context Alone

- Consider a network of friendships for which no user-item rating matrix is provided.

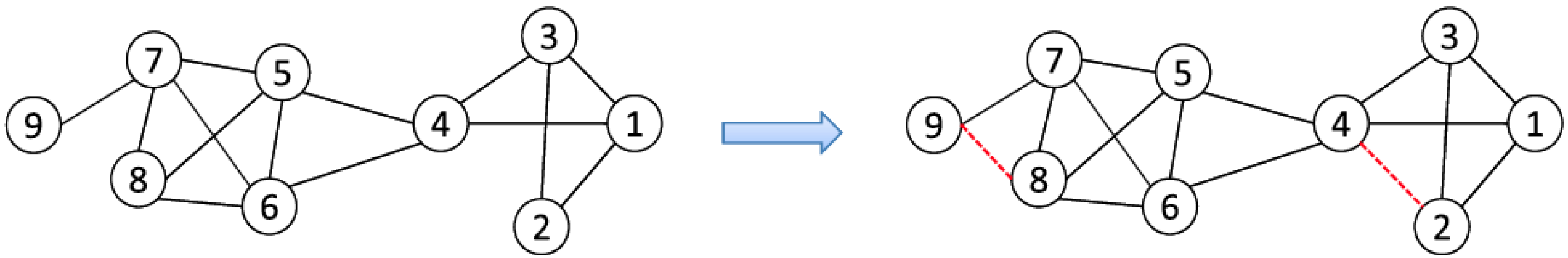
- In this network, we can still recommend users from the network to other users for friendship.

- This is an example of friend recommendation in social networks.



Recommendation Using Social Context Alone

- Very common in social media applications
  - Tag, Friend, Group, Media, Link Recommendations



Link prediction

Recommender Systems: Facebook

- ❖ Facebook friend recommendations
- ❖ “People you may know”
- ❖ “Based on mutual friends, work and education information, networks you’re part of, contacts and many other factors.”
- ❖ “Since our formula is automatic, you might occasionally see people you don’t know or don’t want to be friends with. To remove them from view, just click the X next to their names.”

http://www.facebook.com/help/?page=199421896769556

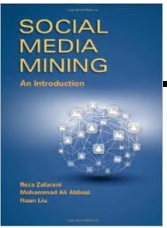
Extending Classical Methods

- Using **social information** in addition to a **user-item rating matrix** to improve recommendation.
- See Textbook #2 for an example of such a model:

$$\min_{U,V} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - U_i^T V_j)^2 + \beta \sum_{i=1}^{n}\sum_{j\in F(i)} sim(i,j)\|U_i - U_j\|_F^2$$
$$+ \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2$$

$R \in \mathbb{R}^{n\times m}$, $U \in \mathbb{R}^{k\times n}$, $V \in \mathbb{R}^{k\times m}$

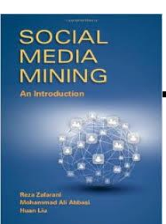
F(i) is friends of user *i*
sim(i, j) is the similarity between users *i* & *j*



Recommendation Constrained Social Context

- In classical recommendation,
  - To estimate ratings, we determine similar users or items.
  - Any user similar to the individual can contribute to the predicted ratings for the individual.
- We can limit the set of individuals that can contribute to the ratings of a user to the set of friends of the user.
  - $S(i)$ is the set of k most similar **friends** of an individual

$$r_{u,i} = \bar{r}_u + \frac{\sum_{v \in S(u)} sim(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in S(u)} sim(u,v)}$$



Recommendation Constrained Social Context

$$A = \begin{array}{c} \\ \mathit{John} \\ \mathit{Joe} \\ \mathit{Jill} \\ \mathit{Jane} \\ \mathit{Jorge} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathit{John} & \mathit{Joe} & \mathit{Jill} & \mathit{Jane} & \mathit{Jorge} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

	Lion King	Aladdin	Mulan	Anastasia
John	4	3	2	2
Joe	5	2	1	5
Jill	2	5	?	0
Jane	1	3	4	3
Jorge	3	1	1	2

Average Ratings

$$\bar{r}_{John} = \frac{4+3+2+2}{4} = 2.75.$$

$$\bar{r}_{Joe} = \frac{5+2+1+5}{4} = 3.25.$$

$$\bar{r}_{Jill} = \frac{2+5+0}{3} = 2.33.$$

$$\bar{r}_{Jane} = \frac{1+3+4+3}{4} = 2.75.$$

$$\bar{r}_{Jorge} = \frac{3+1+1+2}{4} = 1.75.$$

User Similarity

$$sim(Jill, John) = \frac{2\times 4 + 5\times 3 + 0\times 2}{\sqrt{29}\sqrt{29}} = 0.79$$

$$sim(Jill, Joe) = \frac{2\times 5 + 5\times 2 + 0\times 5}{\sqrt{29}\sqrt{54}} = 0.50$$

$$sim(Jill, Jane) = \frac{2\times 1 + 5\times 3 + 0\times 3}{\sqrt{29}\sqrt{19}} = 0.72$$

$$sim(Jill, Jorge) = \frac{2\times 3 + 5\times 1 + 0\times 2}{\sqrt{29}\sqrt{14}} = 0.54$$

$$\begin{aligned} r_{Jill,Mulan} &= \bar{r}_{Jill} + \frac{sim(Jill,Jane)(r_{Jane,Mulan} - \bar{r}_{Jane})}{sim(Jill,Jane) + sim(Jill,Jorge)} \\ &\quad + \frac{sim(Jill,Jorge)(r_{Jorge,Mulan} - \bar{r}_{Jorge})}{sim(Jill,Jane) + sim(Jill,Jorge)} \\ &= 2.33 + \frac{0.72(4-2.75) + 0.54(1-1.75)}{0.72+0.54} = 2.72 \end{aligned}$$