Popularity

Recommendation System

Agenda

- Recommendation systems overview
- Types of recommendation systems
- Popularity based recommendation systems
- Content based recommendation systems
- Similarity fundamentals

Recommendation Systems (TOC)

S. No.	Topic	Scope	Objective
1	Introduction to recommendation systems	Discuss what recommendation system is and why is it imp	To understand the motivation and need for recommendation systems
2	Types of recommendation systems	Discussing various types of recommendation system	To understand various methods to build a recommender
3	Popularity based recommender	Discuss theory behind popularity based model	How to recommend popular items to a new user
5	Content based model	Advantages, BOW, TF	To understand how to recommend items using content based
6	Similarity fundamentals	What and why of similarity functions, different types, their application, Euclidean, Jaccard, Pearson, Cosine	To understand the fundamental of similarity, to be able to use write method to calculate similarity distance

Overview

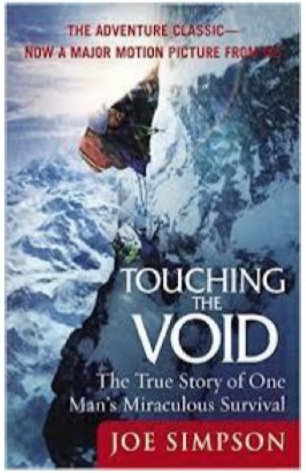
- Provides most relevant item/information to a user
- Uses pattern from the existing data
- Build using popularity and similarity concepts

The recommendation problem

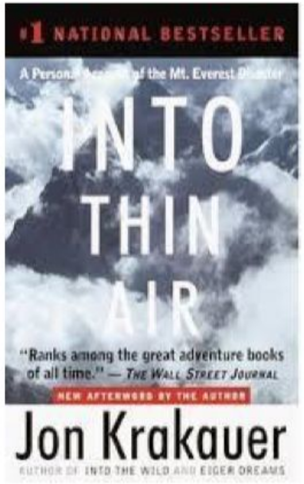
- The recommendation problem can be defined as estimating the response of a user for new items, based on historical information stored in the system, and suggesting to this user novel and original items for which the predicted response is high. The type of user-item responses varies from one application to the next, and falls in one of three categories: scalar, binary and unary.

- Scalar responses, also known as ratings, are numerical (e.g., 1-5 stars)
- ordinal (e.g., strongly agree, agree, neutral, disagree, strongly disagree) values representing the possible levels of appreciation of users for items.
- Binary responses, on the other hand, only have two possible values encoding opposite levels of appreciation (e.g., like/dislike or interested/not interested).
- Unary responses capture the interaction of a user with an item (e.g., purchase, online access, etc.) without giving explicit information on the appreciation of the user for this item. Since most users tend to interact with items that they find interesting, unary responses still provide useful information on the preferences of users.

Recommendation systems classical example


THE ADVENTURE CLASSIC—
NOW A MAJOR MOTION PICTURE
TOUCHING THE VOID
The True Story of One Man's Miraculous Survival
JOE SIMPSON

Published in 1988


#1 NATIONAL BESTSELLER
INTO THIN AIR
"Ranks among the great adventure books of all time." — The Wall Street Journal
NEW AFTERWORD BY THE AUTHOR
Jon Krakauer
AUTHOR OF INTO THE WILD AND EIGER DREAMS

Published in 1996

In 1988, a British mountain climber named Joe Simpson wrote a book called Touching the Void, a harrowing account of near death in the Peruvian Andes. It got good reviews, only a modest success, it was soon forgotten. Then, a decade later, a strange thing happened. Jon Krakauer wrote Into Thin Air, another book about a mountain-climbing tragedy, which became a publishing sensation. Suddently, Touching the Void started to sell again."...The Long Tail by Chris Anderson

Recommendation System Applications

- Netflix recommends movies
- E-commerce sites recommends item based on our purchase history
- Youtube recommends videos
- Song recommendations by spotify
- Food recommendations by restaurants
- Social media sites recommends content
- Courses in e-learning
- Jobs
- Advertising messages

Why Recommendation Systems ?

- Increase in sales by personalizing offers
- Enhanced customer experience
- More time spent on platform
- Helps user to find item of their interest
- Helps provider to deliver an item to the right user

Does it work?

NETFLIX 2/3 Rented Movie are from recommendation

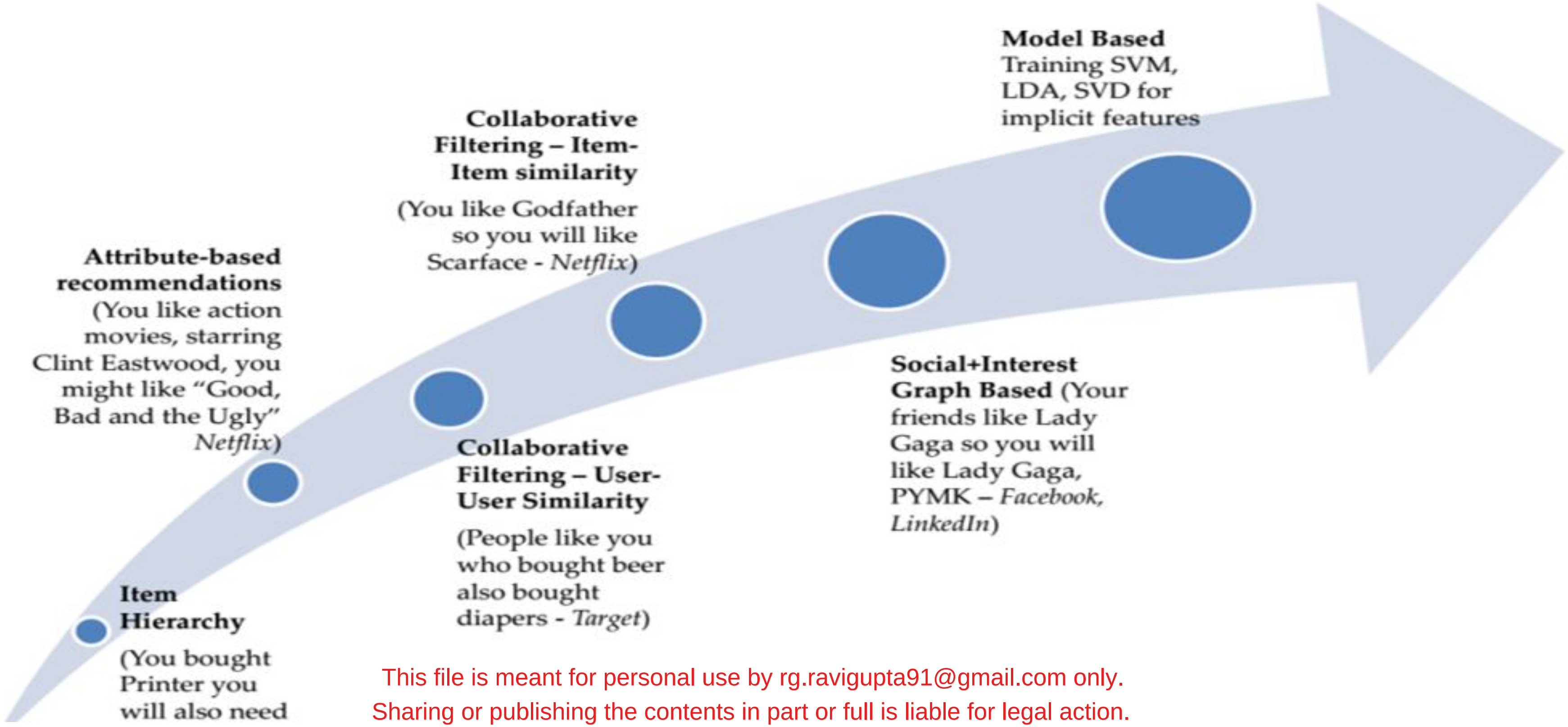
Google news 38% more Click through are due to recommendation.

amazon 35% sales are from recommendation.

Uses

1. Prediction - Estimate of rating a user would assign
2. Recommendation - List of items for user to consider
3. Classification - to predict whether a user will like a product or not using features of both users and products

Evolution to present


Item Hierarchy
(You bought Printer you will also need
Attribute-based recommendations (You like action movies, starring Clint Eastwood, you might like "Good, Bad and the Ugly" *Netflix*)
Collaborative Filtering – User-User Similarity
(People like you who bought beer also bought diapers - *Target*)
Collaborative Filtering – Item-Item similarity
(You like Godfather so you will like Scarface - *Netflix*)
Social+Interest Graph Based (Your friends like Lady Gaga so you will like Lady Gaga, PYMK – *Facebook, LinkedIn*)
Model Based Training SVM, LDA, SVD for implicit features

Types of recommendation systems

- Popularity based
- Content based
- Collaborative Filtering
  - User based
  - Item based
- Matrix factorization
- Association Rule
- Hybrid

Popularity based

- Recommends items which are popular/purchased/rated by most users.

Example: Popular news articles

- It uses:
    - Context
    - Purchase history of other users
    - User-Item features - ratings of other users
- Recommendations are non-personalized
- Every user gets the same recommendations irrespective of their taste and preferences.
- Works with the trend
- Does not suffer from cold start problem

Steps involved for a popularity based model

1. Read the dataset
2. Use group by or SQL merge concept to get the ratings for each item/movie
3. Take mean of all the item/movie
4. Sort it in descending order
5. Recommend top ten or five to a new user

Notes:

- Takes only mean into consideration
- User bias can be removed by deducting mean rating of a user for each item.

Hands-on

Case study

Content based

Recommendation System

Content Based Model

- Recommendations are based on information on the content of items rather than on other user’s opinions

Idea

- If you like an item then you will also like a **similar** item
- Recommend items to user **x** similar to previous items rated highly by **x**

- No need for the data on other users
- No cold start and sparsity Problem
- Technique that can be used- **cosine similarity**

Content representation

- Items that can be recommended to the user are represented by a set of features,also called attributes or properties. For example, in a movie recommendation application, features adopted to describe a movie are: actors, directors, genres, subjectmatter, etc.
- Item descriptions can be textual features extracted from Web pages, emails, news articles or product descriptions. Unlike structured data, there are no attributes with well-defined values. Textual features create a number of complications when learning a user profile, due to the natural language ambiguity

Content Based Model

- A content based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link).
- Based on that data, a user profile is generated, which is then used to make suggestions to the user.
- As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.
- The information source that content-based filtering systems are mostly used with are text documents. A standard approach for term parsing selects single words from documents.

Types of ratings from User

Explicit ratings :
Users rate for an item
Most accurate description of a user's preference
Challenging in collecting data

Implicit ratings:
Observation of user behaviour
Can be collected with less cost to user
Ratings inference may not be precise

Explicit rating from users

- There are three main approaches to get explicit relevance feedback:
- **like/dislike** – items are classified as “relevant” or “not relevant” by adopting a simple binary rating scale.
- **ratings –** a discrete numeric scale is usually adopted to judge items, such as in Alternatively, symbolic ratings are mapped to a numeric scale, where users have the possibility of rating a Web page as hot, lukewarm, or cold;
- **text comments** – Comments about a single item are collected and presented to the users as a means of facilitating the decision-making process. For instance, customer’s feedback at Amazon.com or eBay.com might help users in deciding whether an item has been appreciated by the community. Textual comments are helpful, but they can overload the active user because she must read and interpret each comment to decide if it is positive or negative.

Components of content based recommendation systems

- The high level architecture of a content- based recommendation process is performed in three steps, each of which is handled by a separate component:
- **Content Analyzer:** When information has no structure (e.g. text), some kind of pre-processing step is needed to extract structured relevant information. Data items are analyzed by feature extraction techniques in order to shift item representation from the original information space to the target one
- **Profile Learner:** This module collects data representative of the user preferences and tries to generalize this data, in order to construct the user profile.
- **Filtering component :** This module exploits the user profile to suggest relevant items by matching the profile representation against that of items to be recommended.

User Profiles

- **A model of the user's preferences, i.e., a description of the types of items that interest the user.** There are many possible alternative representations of this description, but one common representation is a function that for any item predicts the likelihood that the user is interested in that item. For efficiency purposes, this function may be used to retrieve the n items most likely to be of interest to the user.

- **A history of the user's interactions with the recommendation system**. This may include storing the items that a user has viewed together with other information about the user's interaction, (e.g., whether the user has purchased the item or a rating that the user has given the item). Other types of history include saving queries typed by the user (e.g., that a user searched for an Italian restaurant in the 90210 zip code).

Implementation of a content based model

1. Read data
2. Preprocessing
3. Distance matrix calculation
4. Check similarity
5. Recommend

Pre-processing

- Remove noise
- Missing values handling
- Get the features needed for the model, align to the items
- Categorical encoding (count vectorizer , One hot encoding)
- Creating the User item rating matrix

Rating Matrix creation from documents

Texts are converted to a document term matrix using cosine similarity

Document 1- Pride of India : A glimpse into India’s scientific heritage

Document 2- Makers of modern India

Document 3 - The logic of scientific discovery

Test doc- Scientific India

Recommendation order would be - D1, D2/D3

Term Frequency

D1 - Machine Learning is fun

Document	Machine	Learning	is	fun
TF	1	1	1	1
Normalized TF	0.25	0.25	0.25	0.25

- Similarly we need to create normalized TF for each document
- Calculate cosine similarity between test doc and existing docs
- Recommend based on similarity score

Similarity fundamentals

- **Euclidean distance**

$$D_{euclidean} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \;\ldots\ldots.\,(x_n - y_n)^2}$$

  - Does not efficiently use for comparisons

- **Cosine Similarity**

$$D_{cosine} = \frac{\sum_i x_i . y_i}{||x||\,||y||}$$

  - Similarity is the cosine of the angle between the 2 vectors.
  - Closer the vectors, smaller will be the angle and larger the cosine.

Similarity fundamentals

- **Pearson Similarity**

$$D_{pearson} = \frac{\sum_i (x_i - \overline{x}).(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2 \sum_i (y_i - \overline{y})^2}}$$

  - Similarity is the pearson coefficient between the two vectors
  - Pearson correlation is invariant to shift.
  - Pearson and cosine similarity are invariant to scaling
  - The range of similarity varies between -1 to 1.
- **Jaccard Similarity**

$$D_{jaccard} = \frac{|A \cap B|}{|A \cup B|}$$

  - used to comparing the similarity and diversity of sample set.

Algorithm: Nearest Neighbors Method

- The nearest neighbor algorithm simply stores all of its training data, here textual descriptions of implicitly or explicitly labeled items, in memory.
-
- In order to classify a new, unlabeled item, the algorithm compares it to all stored items using a similarity function and determines the "nearest neighbor" or the k nearest neighbors.
- The class label or numeric score for a previously unseen item can then be derived from the class labels of the nearest neighbors.

Similarity functions used by the nearest neighbor algorithm

The similarity function used by the nearest neighbor algorithm depends on the type of data.

- For **structured data**, a Euclidean distance metric is often used. In the Euclidean distance function, the same feature having a small value in two examples is treated the same as that feature having a large value in both examples.
- When using the **vector space mode**l, the cosine similarity measure is often used. the cosine similarity function will not have a large value if corresponding features of two examples have small values.
- Cosine similarity is appropriate for text when we want two documents to be similar when they are about the same topic, but not when they are both not about a topic.

Advantages of content based model

- Content-based recommenders exploit solely ratings provided by the active user to build her own profile.
- Content-based recommenders are capable of recommending items not yet rated by any user. As a consequence, they do not suffer from the first-rater (cold start) problem.
- Explanations on how the recommender system works can be provided by explicitly listing content features or descriptions that caused an item to occur in the list of recommendations.

Disadvantages of content based model

- Content-based systems suffer from **over-specialization**, since they recommend only items similar to those already rated by users. One possible solution to address this problem is the introduction of some randomness.
- Content-based techniques have a natural **limit in the number and type of features** that are associated, whether automatically or manually, with the objects they recommend. Domain knowledge is often needed, e.g., for movie recommendations the system needs to know the actors and di- rectors, and sometimes, domain ontologies are also needed.
- **Enough ratings** have to be collected before a content-based recommender system can really understand user preferences and provide accurate recommendations.

Hands-on

Case study

Collaborative Filtering

Recommendation System

Agenda

- What is collaborative filtering
- Types of collaborative filtering
- User-user, item-item
- Challenges
- Steps
- Matrix Factorization - SVD
- SVD for collaborative filtering
- Surprise library
- Hands-on

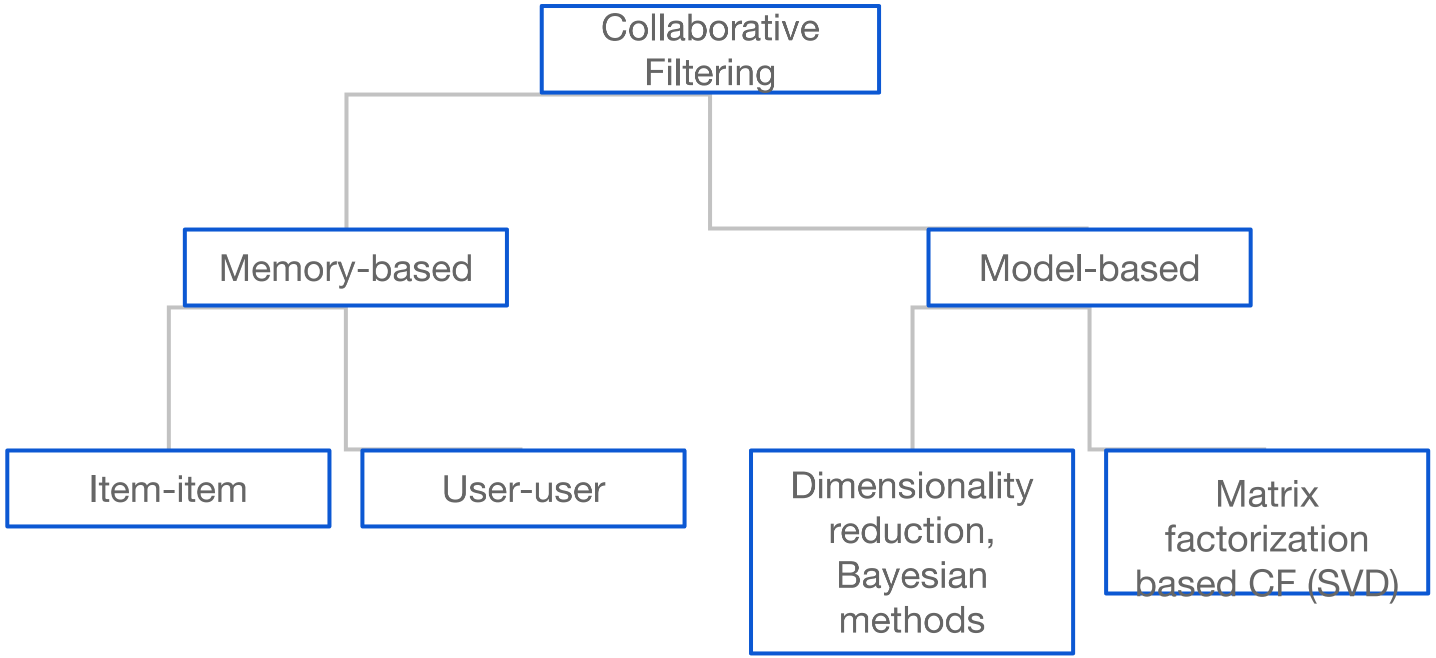
Recommendation systems (TOC)

S. No.	Topic	Scope	Objective
1	What is collaborative filtering	Understand collaborative filtering, recommendation using collaborative models	To be able to define collaborative filtering, understand the theory behind it
2	Types of collaborative filtering	Model based, memory based, Item-item, user-user	To discuss various ways to build a collaborative filtering model
3	Steps	Discuss major steps involved in CF, similar user, rating calculation, evaluation metrics	
4	Matrix Factorization	Matrix factorization, different techniques like SVD, NMF, SVD for recommendation	To utilize svd for building recommendation systems
5	Surprise library	How to deal with surprise dataset, SVD, KNNWithMeans, uid, iid,	To be able to work with surprise library

Collaborative Filtering

- Idea is to find similarity between users and items.
- Uses user behavior for recommendation
- Algorithm is based on the past behavior and not on context
- Data contains set of users and items and rating/reaction
- Make use of rating rating matrix to find similar users

Types of Collaborative Filtering


Collaborative Filtering
Memory-based
Model-based
Item-item
User-user
Dimensionality reduction, Bayesian methods
Matrix factorization based CF (SVD)

Collaborative Filtering

Memory based	Model Based
● Similarity between users and/or items are calculated and used as weights to predict a rating ● Memory based approaches directly works with values of recorded interactions, assuming no model, and are essentially based on nearest neighbours search Eg - Item based, User based	● Model based approaches assume an underlying "generative" model that explains the user-item interactions and try to discover it in order to make new predictions. ● Can solve problem of huge database & sparse matrix ● Better at dealing with sparsity ● Predicts ratings of unrated items ● Inference is not traceable due to hidden factors Eg-Matrix factorization (SVD, PMF, NMF), Neural nets based

Steps for Memory based CF

1. Determine similar users

2. a. Every user can be represented by its vector of interactions with the different items. Calculate similarity matrix using similarity distance and user-item ratings. Get top similar neighbors

3. Estimate rating that a user would give to an item based on the ratings of similar users

   a. Estimated rating R for a user U for an item I would be close to average rating given to I by the top n users most similar to U

   b. R_u = $(\sum^{n}_{u=1}$R_u)/n

   c. Weighted average - multiply each rating by similarity factor (Rating Normalization)

Note - user bias can be removed by subtracting mean rating given by that user to all the items for each item

Dataset - rating matrix

Matrix with mostly empty cells is called **sparse**

the opposite to that (a mostly filled matrix) is called **dense**.

These interactions are stored in the so-called “user-item interactions matrix”.

Rating Normalization

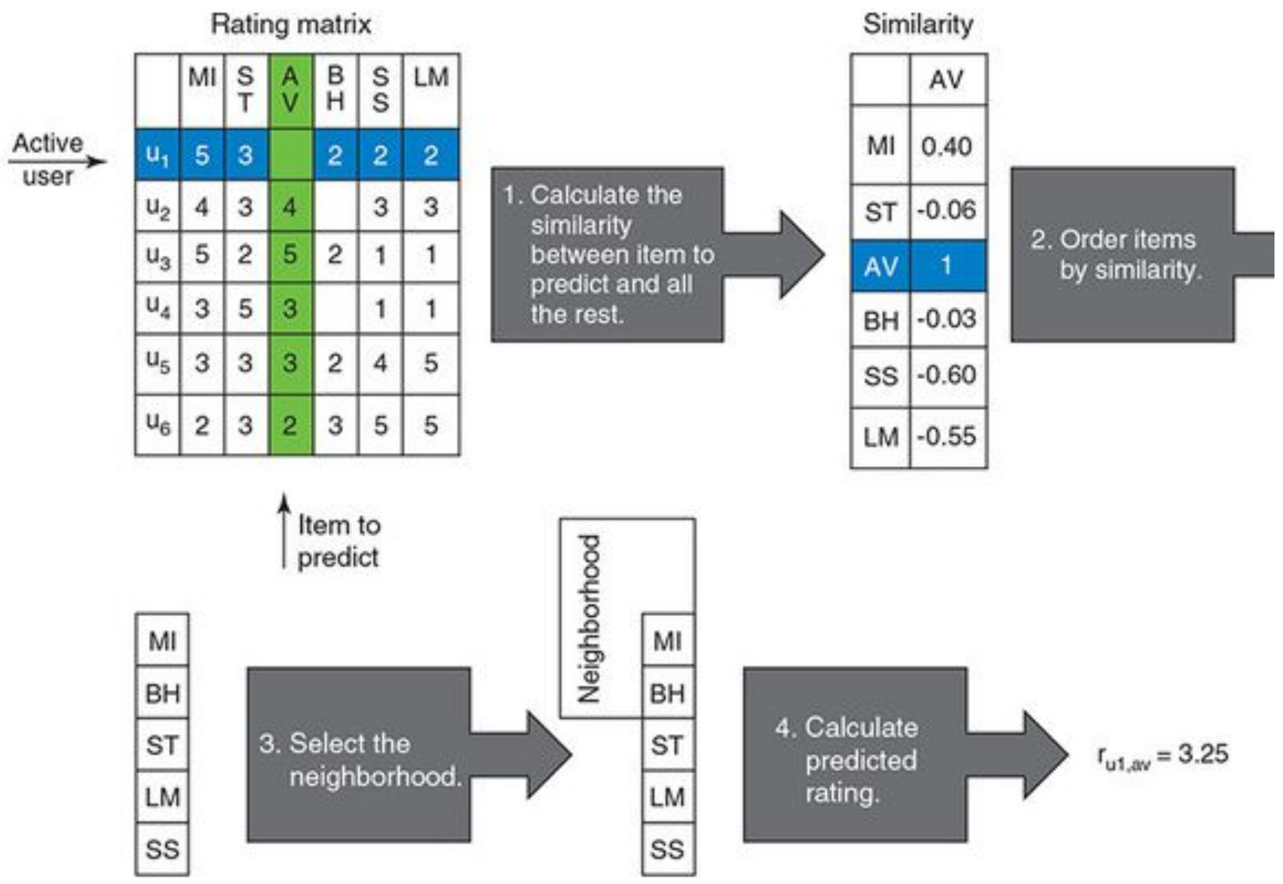
- When it comes to assigning a rating to an item, each user has its own personal scale. Even if an explicit definition of each of the possible ratings is supplied (e.g., 1=“strongly disagree”, 2=“disagree”, 3=“neutral”, etc.).

- Two of the most popular rating normalization schemes that have been proposed to convert individual ratings to a more universal scale are **mean-centering** and **Z-score**.

Collaborative Filtering

Item based - similarity between each pair of items is calculated.

- Neighboring items are considered
- Let’s say item X and Y are purchased together, and if someone is buying X then Y will be recommended to him.
- Find similar items to the ones the active user likes
- Calculate predicted ratings for these items
- Use predictions to calculate recommendations



Rating matrix
Active user
Item to predict
1. Calculate the similarity between item to predict and all the rest.
Similarity
2. Order items by similarity.
3. Select the neighborhood.
Neighborhood
4. Calculate predicted rating.
$r_{u1,av} = 3.25$

	MI	ST	AV	BH	SS	LM
u_1	5	3		2	2	2
u_2	4	3	4		3	3
u_3	5	2	5	2	1	1
u_4	3	5	3		1	1
u_5	3	3	3	2	4	5
u_6	2	3	2	3	5	5

	AV
MI	0.40
ST	-0.06
AV	1
BH	-0.03
SS	-0.60
LM	-0.55

MI
BH
ST
LM
SS

Neighborhood
MI
BH
ST
LM
SS

Image credits: Practical recommender systems (book)

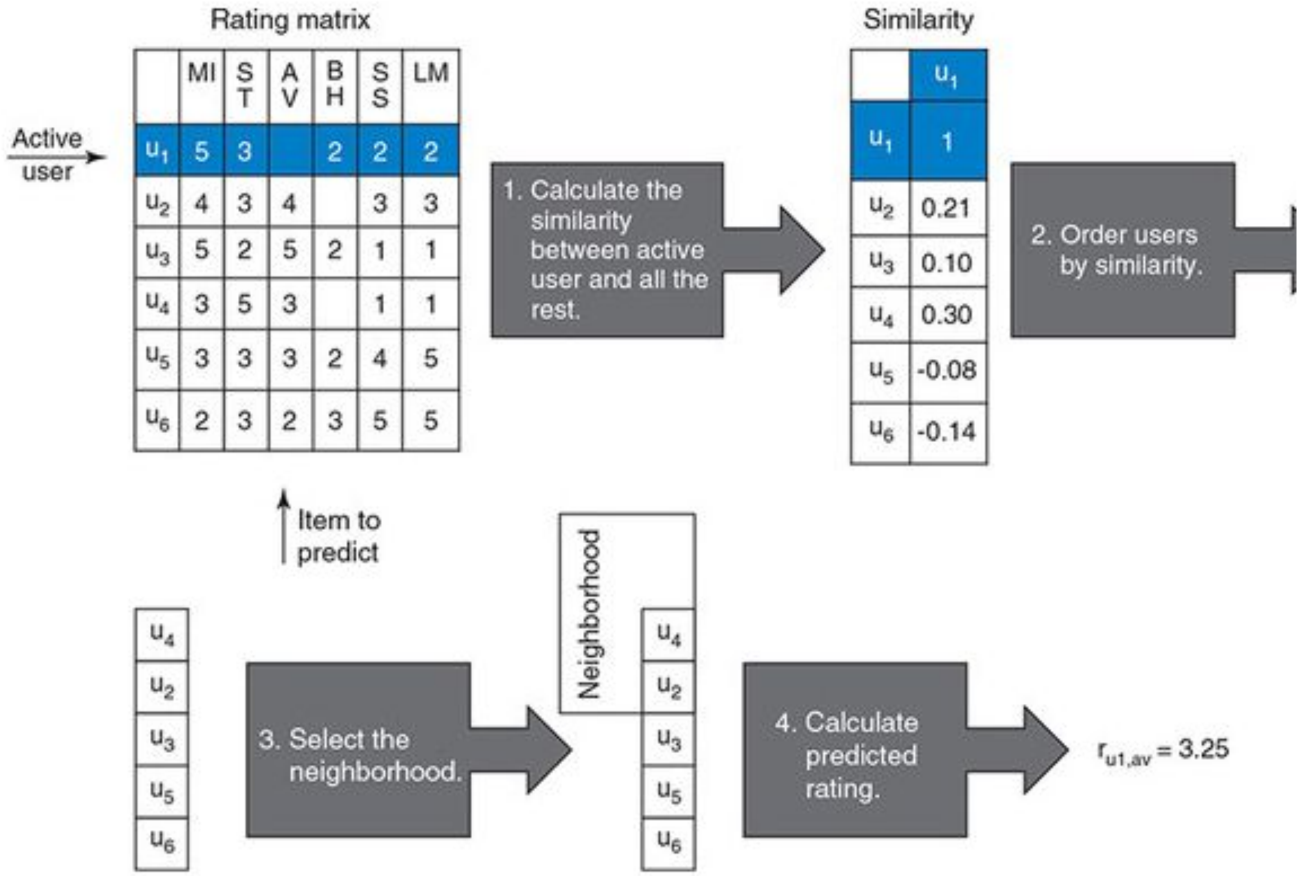
Decoding the formulae

$$Pred(u,i) = \bar{r}_u + \frac{\sum_{j \in S_i} (sim(i,j) \times r_{u,j})}{\sum_{j \in S_i} sim(i,j)}$$

- $\bar{r}_u$ is the average rating of the user u.
- $r_{u,j}$ is the active user's u rating of item j.
- S_i is the set of items in the neighborhood that user u has rated.
- $Pred(u,i)$ is the predicted rating for user u of item i.
- $sim(i,j)$ is the similarity between item i and item j.

User based Methods

- similar users are considered
- Let’s say A and B like same movies, then a new movie liked by A will be recommended to B as well
- User-user method roughly tries to identify users with the most similar “interactions profile” (nearest neighbours) in order to suggest items that are the most popular among these neighbours (and that are “new” to our user)



Rating matrix
Active user
1. Calculate the similarity between active user and all the rest.
Similarity
2. Order users by similarity.
Item to predict
3. Select the neighborhood.
Neighborhood
4. Calculate predicted rating.
$r_{u1,av} = 3.25$

	MI	ST	AV	BH	SS	LM
u_1	5	3		2	2	2
u_2	4	3	4		3	3
u_3	5	2	5	2	1	1
u_4	3	5	3		1	1
u_5	3	3	3	2	4	5
u_6	2	3	2	3	5	5

	u_1
u_1	1
u_2	0.21
u_3	0.10
u_4	0.30
u_5	-0.08
u_6	-0.14

u_4
u_2
u_3
u_5
u_6

Image credits: Practical recommender systems (book)

Decoding the formulae

$$pred(u,i) = \bar{r}_u + \frac{\sum_{n \subset neighbors(u)} sim(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \subset neighbors(u)} sim(u,n)}$$

- *pred(u,i)* is the predicted reting for user *u* to item *i*
- *neighbors(u)* is set of the users similar to the user *u*
- *sim(u,n)* is the similarity score between the active user *u* and the *n*
  _ ighbors

$\bar{r}_u$

- r_{ni} s the average rating of user u
- $\bar{r}_n$ is the rating given by n neighbors to item i

Is the average rating for n neighbors

User based methods vs Item based methods

- The accuracy and efficiency of neighborhood recommendation methods depends mostly on the ratio between the number of users and items in the system.
- similarity between two users in userbased methods, which determines the neighbors of a user, is normally obtained by comparing the ratings made by these users on the same items.
- In cases where the number of users is much greater than the number of items, such as large commercial systems like Amazon.com, item-based methods can therefore produce more accurate recommendations.
- Likewise, systems that have fewer users than items, may benefit more from user-based neighborhood methods
- Item-based recommendation approaches require much less memory and time to compute the similarity weights (training phase) than user- based ones, making them more scalable.

Challenges with CF

- **Cold Start problem:** The model requires enough amount of other users already in the system to find a good match.
- **Sparsity:** If the user/ratings matrix is sparse, and it is not easy to find the users that have rated the same items.
- **Popularity Bias**: Not possible to recommend items to users with unique tastes.
  - Popular items are recommended more to the users as more data being available on them
  - This may begin a positive feedback loop not allowing the model to recommend items with less popularity to the users
- **Shilling attacks**
  - Users can create fake preferences to push their own items

Advantages of memory based techniques

- **Serendipity:** Serendipity extends the concept of novelty by helping a user find an interesting item he might not have otherwise discovered. *For example, recommending to a user a movie directed by his favorite director constitutes a novel recommendation if the user was not aware of that movie, but is likely not serendipitous since the user would have discovered that movie on his own.*

Real time implementation in the state of the art

- **Item-to-Item Collaborative Filtering**
- http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

Collaborative effort between MIT and the University of Minnesota

- https://grouplens.org/

http://ccs.mit.edu/papers/CCSWP165.html

Model based recommendation systems

- Model-based approaches excel at characterizing the preferences of a user with **latent factors**. For example, in a movie recommender system, such methods may determine that a given user is a fan of movies that are both funny and romantic, without having to actually define the notions “funny” and “romantic”
- Singular value decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix.
- In SVD++ Prediction accuracy is improved by considering also implicit feedback, which pro-vides an additional indication of user preferences. This is especially helpful for thoseusers that provided much more implicit feedback than explicit one.

Matrix factorization

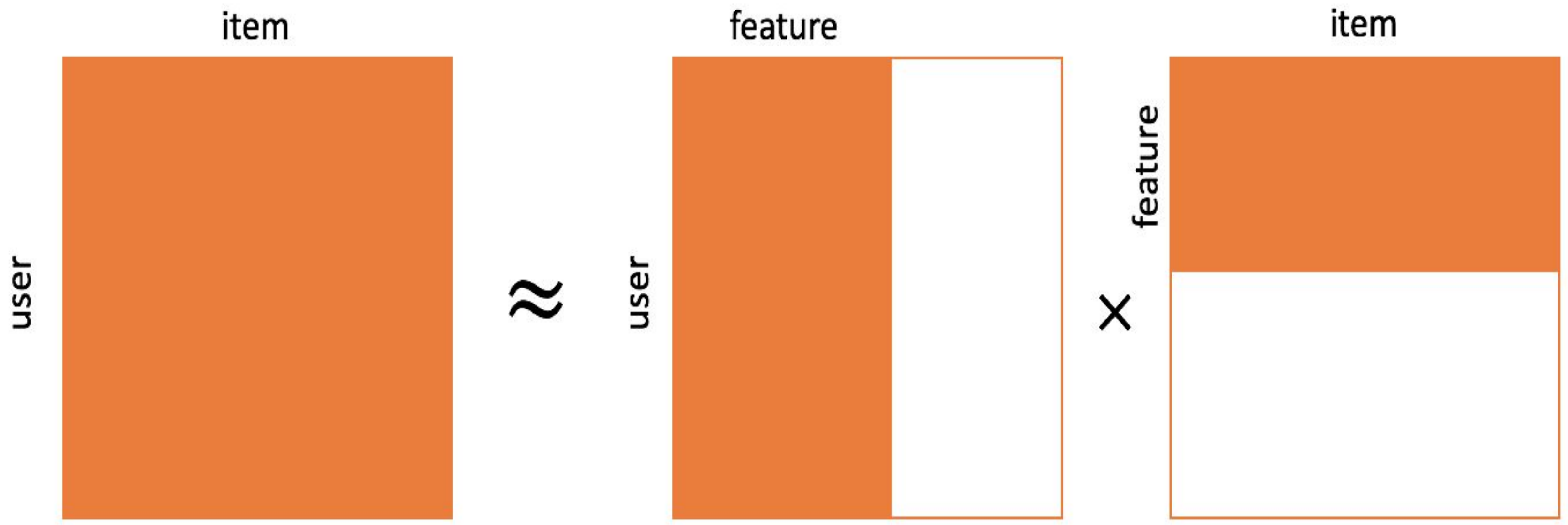
- Idea is to find preferences using some hidden factors
- Idea is to break down a large matrix (user-item) into a product of smaller ones

  Ex - 12 = 6*2, 3*4

  Similarly, a matrix **A** with dimension m*n can be reduced to product of two matrices **X** and **Y** with dimension m*p and p*n respectively. [p should be common]

- Here, m and n matrices represents latent factors
- Techniques
  - SVD - singular value decomposition (orthogonal factorization)
  - PMF - probabilistic factorization

Matrix Factorization



Singular value decomposition

- Given a square or non square matrix A, linear Algebra theorem SVD specifies that: $A_{(m\times n)} = U_{(m\times m)} * S_{(m\times n)} * V_{(n\times n)}^T$

Where,

A is $m\times n$ matrix

U is an $m\times m$ orthogonal matrix

S is a $m\times n$ diagonal matrix- (singular value)

V is a $n\times n$ orthogonal matrix

- The matrices U and V are orthogonal so: $U^T.U=V.V^T=I$

Singular value decomposition

	2		
	1.6		

n matrix (2*4)

m (3*2) matrix

3	1.2

	4		

Matrix A : m*n (3*4)

Challenges for any Recommender systems

- Scalability of the algorithms with large and real-world datasets
- Proactive recommender systems, i.e., recommenders that decide to provide recommendations even if not explicitly requested. The largest majority of the recommender systems developed so far follow a “pull” model;where the user originates the request for a recommendation.
- Privacy preserving recommender systems
- Diversity of the items recommended to a target user
- Integration of long-term and short-term user preferences in the process of building a recommendation list
- Generic user models and cross domain recommender systems are able to mediateuser data through different systems and application domains

Content vs collaborative filtering

- Collaborative approaches overcome some of the limitations of content-based ones.

- Items for which the content is not available or difficult to obtain can still be recommended to users through the feedback of other users.

- Collaborative recommendations are based on the quality of items as evaluated by peers, instead of relying on content that may be a bad indicator of quality.

- Finally, unlike content-based systems, collaborative filtering ones can recommend items with very different content, as long as other users have already shown interest for these different items.

Surprise Library

- It’s a add-on package for scipy. It is hosted and developed separately from main scipy distribution
- Comes with various recommender algorithms and similarity metrics
- How to install -
  - `$ pip install scikit-surprise`
  - `$ conda install -c conda-forge scikit-surprise`
- Common commands
  - `from surprise import Dataset, Reader, SVD, KNNWithMeans`
  - `from surprise.model_selection import GridSearchCV`
- `# Loads Pandas dataframe`

Validation of Recommendation systems

- A recommender model is built against the training set. The users in the test set are then considered in turn, and have their ratings or purchases split into two parts, the query set and the target set.

- The standard method for computing predictive accuracy is mean absolute error (MAE) – the average absolute difference between the predicted rating and the actual rating given by a user.

- Root mean square error (RMSE) is also used very often for computing the models accuracy.

- K-Fold cross validation is also used to check for the consistency of the model

References

- https://surprise.readthedocs.io/en/stable/matrix_factorization.html
- https://surprise.readthedocs.io/en/stable/knn_inspired.html

Hands-on

Case study