Recommender System

- Introduction
  - 1.1. Recommender Systems Tasks
  - 1.2. Recommender Systems Types
  - 1.3. Recommender Systems Scope
  - 1.4. Recommender Systems Benefits
- Paradigms of Recommendation Systems
  - 2.1. Personalized Recommendation
  - 2.2. Collaborative Filtering Recommendation
  - 2.3. Content-based Recommendation
  - 2.4. Knowledge-based Recommendation
  - 2.5. Hybrid Recommendation
- Collaborative Filtering
  - 3.1. User-Based Nearest Neighbor
  - 3.2. Item-Based Nearest Neighbor
  - 3.3. Measuring User Similarity: Pearson Correlation
  - 3.4. Collaborative Filtering
- Association Rule Mining
  - 4.1. Support
  - 4.2. Confidence
  - 4.3. Lift
- Association Rule Market Basket Analysis
- Recommender Systems
  - 5.1. Recommender Systems Types
  - 5.2. Recommender Systems Utility
  - 5.3. Recommender Systems Scope
  - 5.4. Recommender Systems Benefits
- Performance Measures
  - 6.1. Mean Absolute Error (MAE)
  - 6.2. Root Mean Square Error (RMSE)
  - 6.3. Precision
  - 6.4. Recall
  - 6.5. F-Measure
- The Cosine and Adjusted Cosine Similarity Measures
  - 7.1. Cosine Similarity
  - 7.2. Adjusted Cosine Similarity
- Association Rule Mining
  - 8.1. Support
  - 8.2. Confidence
  - 8.3. Lift
- Association Rule Market Basket Analysis

Introduction

Many E-commerce and smart companies implement recommender systems on their websites, to leverage the power of data and provide their users, in either audio or visual, the most relevant information to guide or persuade them and accomplish the main uses may be: Recommendation, Personalization, Ranking, Explanation.

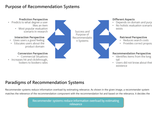
Recommender system is an information filtering technique, which provides users with recommendations, which they can might be interested in.

Recommender Systems Solution

Recommender Systems act as a solution for your day-to-day choices.

Recommender Systems Features

Recommender Systems: Example



Purpose of Recommendation Systems

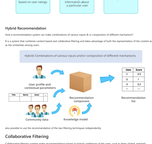
Paradigms of Recommendation Systems

Recommender systems can be implemented using several different approaches. As shown in the given image, a recommendation system matches the relevance of the recommendation component with the recommendation list and presents it to the interested. It decides the

Personalized Recommendation

The most relevant item is recommended based on the user profile and contextual parameters. Everyday recipe products of information and personalization are the existence of a large amount of data, which makes it difficult to find the required information.

Personalized recommendation is a process of alleviating the problem.



Collaborative Filtering Recommendation

Let's analyze how a recommendation system works. Do you like people among your peers?

If yes the same idea is actually the same for similarity between items. It does not consider the contextual relationship between different items.

Content-based Recommendation

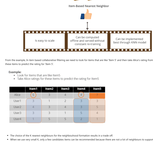
It is one to display products similar to the products we have liked before.

In content-based recommender system, only the data provided by the user either explicitly through rating or implicitly by clicking on a link. Based on it, data provided by the user. It generates a user profile suggestion and gain to use based on this.

Knowledge Issue Recommendation

Recommender the product based on user requirements.

Knowledge-based recommender system. Recommends the products based on user's requirements.



Hybrid Recommendation

Hybrid recommendation system are made combinations of various issues & or comparison of different methods!

It is a system that combines recommendations of various approaches and take advantage of both the recommendations of the content and and the collaborative analysis.

Collaborative Filtering

Collaborative filtering system make recommendations based on historic preference of the users, such as likes, clicked, watched, purchased, ratings.

Item based users and user based users are two different approaches.

- User based nearest neighbor
- Item based nearest neighbor

User-Based Nearest Neighbor

Measuring User Similarity: Pearson Correlation

However, when the relationship between the variables are not linear, or the data are recorded nominally rank based or ordinal, Pearson's correlation is determine the strength and direction of a linear relationship.

To find the similarity between 2 users a and b, we can use a slang method formula.

The given formula will give us the result between -1 and 1.

where,

1. $r_{a,b}$ means Similarity
2. $r_{a,b}$ ranges from -1 to 1

Item Based Nearest Neighbor

Item based collaborative filtering recommendation algorithm and item based approach is often harder to scale on the items usually don't change much, thus item based algorithm offer better scalability than user based algorithms.

- Item based approach are easy to scale, since the similarity between items don't ever vary or make predictions.
- Item based approach offer a better scalability than user based approach, since the number of items are less than users.
- It is also very good formula for recommender system because

From the example in the chapter, whether Alice will like or not dislike the item 5, we'll find item 1 and item 4 are liked by users and these items is similar to the rating for item 5.

Example:

- Item 1 is similar to item 5
- Item 4 is similar to item 5

- The choice of the K-nearest neighbors for the neighborhood formation leads to a trade-off.
- If the neighborhood is too small, the recommender system may not be recommended because there are not enough of neighbors to support the recommendation.
- While the participation of user preferences can be thought due to the large neighborhood size. So a large large K, more reasons to recommend.
- The optimum size can be found out by cross-validation and it is typical, for K, to be in the range of around from 5 to 100.

The Cosine and Adjusted Cosine Similarity Measures

Similarity is measured based on two algorithms: cosine and adjusted cosine.

Association Rule Mining

Association Rule Mining also known as Market basket analysis is used to find items that are frequently bought together.

It defines associations rule similarity to identify frequent if-then associations.

Performance Measures

Association rules are created by.

- Searching data for frequent if-then patterns
- Applying necessary criteria
- Identify the most important relationships.

Performance measures for association rule include

- Support
- Confidence
- Lift

Support

Confidence

Lift

Association Rule Market Basket Analysis

Suppose there are four transactions in ITS (ITS={T1,T2,T3,T4}) as given below:

Consider we made some calculations using the given transaction database. Then Support, Confidence and Lift will be:

Rule	Support (How much the historical data supports the association rule?)	Confidence (How confident we are that the association rule is valid?)	Lift (Confidence - Support)
A=>B	Fraction of total cases in which A and B occur together	Fraction of cases in which we can see B when we have A	Confidence - Support
A=>C	Fraction of total cases in which A and C occur together	Fraction of cases in which we can see C when we have A	
A=>D			
B and C=>D			

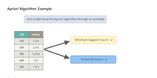
- Support is the measure of how much the historical data supports some defined rule/ confidence mean. It can be calculated as the fraction of cases containing both A and C out of total cardinality of A and C.
- Confidence mean is a measure of how confident we are as the domain expert. Confidence mean how confident we are that if we have A then we can see C in the transaction. It can be calculated as the fraction of cases in which A and C occur together out of the total cases in which we have A.
- Lift is the ratio of the Confidence and Support. It can be calculated as the ratio of Confidence to Support.
- Lift = 1 means Confidence and Support are independent.
- Lift > 1 means Confidence and Support are positively related.
- Lift < 1 means Confidence and Support are negatively related.

So what we are calculating: support, confidence and lift. Lets start with A=>B in the below given table:

Rule	Support	Confidence	Lift
A=>B	3/4	3/3	3/3
A=>C	2/4	2/3	1.6
A=>D	2/4	2/3	1.6
B and C=>D	1/4	1/2	1.6

Association Rule Generation: Apriori Algorithm

Consider one rule: Apriori algorithm



Apriori Algorithm Example

Let's understand the Apriori algorithm through an example.

TID	Items
100	134
200	1235
300	1235
400	1245

Step 1:

The number of transaction that have to be examined can be pruned using the apriori principle. The list of popular item sets can also eliminated.

We have the given data of a store. Take the transaction database to get the support. If item sets have min. Support = 2, then you can eliminate the item sets from the table.

Let's assume the support value.

- Create the item of size 1.
- Take the transaction table. Take the first item 1 support value of 1, which is less than the total support value 2. So discard item 4 in upcoming iteration, as it has the lesser support value.

Apriori Algorithm: Step 01

List of one-item itemsets you made and their support values are calculated.

Since, the threshold value is 2, any itemset with support less than 2 are omitted

Step 2

- Create item sets combining pair a single item.
- Remove the Support for the item sets.
- Remove the Itemset that do not meet our minimum support threshold and keep the item sets that meet it.
- Take the transaction table and count the support.
- Only return pairs in which the combined support meets the threshold.
- Take the item sets that have the support value greater than the threshold value. Item sets that do not meet the threshold, item sets that have support less than 2 are eliminated. Again item sets in 3 are pruned because they have support value less than 2.

Apriori Algorithm: Step 02

The length of the itemset is combined with 2 (1,2,3,4) (1,2,3,4)

Step 3

- Take the transaction table and count the support value.
- Only return pairs in which the combined support meets the threshold.
- Take the item sets that have the support value greater than the threshold value. Item sets that do not meet the threshold, item sets that have support less than 2 are eliminated.

Apriori Algorithm: Step 03

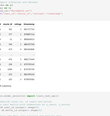
The length of the itemset is combined with 3 (1,2,3,4) (1,2,3,4) (1,2,3,4) new candidate key (1,2,3)

Apriori Algorithm: Step 04

The length of the itemset is combined with 4 (1,2,3,4) (1,2,3,4) (1,2,3,4) (1,2,3,4) new candidate key (1,2,3,4)

Apriori Algorithm: Step 05

Create your itemset to check if there are any other subsets whose support are greater than 2.



Step 6

Take the transaction database as the support. If item sets in transaction table are equal to the given set, compare it with item sets and get a set of frequent item sets, being same as in step 5.

Apriori Algorithm: Step 05

Using the formula of (T10, 2) since Itemset (T4) is invalid.

Since, support of (T10 is less than 2, so self-stop and return to the previous Itemset. (T8)

Step 8

With F(1) we get the following list:

- F(1) = {{1}, {2}, {3}, {4}, {1,2}, {1,3}, {1,5}, {2,3}, {2,5}}
- F(1) = {{1,2,3}, {1,2,5}, {1,3,5}, {2,3,5}}

so with items set we apply the rule on item set F(1)

we have minimum confidence value of 80%.

For every subset S of L we have:

- S = {1,2,3} is mentioned in F(1)
- P(subset)/P = support(S) / min_sup = 2/2

Apriori Algorithm: Step 06

Now, you will prevent self-subset creation with the disjoined list of frequent itemsets.

Confidence Value - 80%

Consider the set generated all such subsets for each frequent itemset
- For L = {1,2,3}, subsets are {1,2}, {1,3}, {2,3}, {1}, {2}, {3}
- For L = {1,2,5}, subsets are {1,2}, {1,5}, {2,5}, {1}, {2}, {5}

For every subset S of L

- P = {1,2,3} is mentioned in F1
- P(subset)/P = min_sup / support(S)

Objective: Predict a book review recommendation model

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
ratings_df.head()
```

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

```
ratings_df.shape
```

```
(100000, 4)
```

```
ratings_df.describe()
```

	user_id	item_id	rating	timestamp
count	100000.00000	100000.00000	100000.00000	1.000000e+05
mean	462.48475	425.53130	3.529860	8.835289e+08
std	266.61442	330.798356	1.125674	5.343856e+06
min	1.000000	1.000000	1.000000	8.747247e+08
25%	254.000000	175.000000	3.000000	8.794487e+08
50%	447.000000	322.000000	4.000000	8.828269e+08
75%	682.000000	631.000000	4.000000	8.882601e+08
max	943.000000	1682.000000	5.000000	8.893848e+08

```
# Create a frequency table for each movie
movie_stats = ratings_df.groupby('item_id').agg({'rating': [np.size, np.sum]})
movie_stats.head()
```

item_id	rating	
	size	sum
1	452	1756
2	131	372
3	90	267
4	192	629
5	59	148

```
# Create a threshold for minimum number of ratings
min_ratings = 50
popular_movies = movie_stats[movie_stats['rating']['size'] >= min_ratings]
popular_movies.head()
```

item_id	rating	
	size	sum
1	452	1756
2	131	372
3	90	267
4	192	629
5	59	148

```
popular_movies.shape
```

```
(1682, 2)
```

```
# Join with the movie titles to get more information
popular_movies = pd.merge(popular_movies, movies_df)
popular_movies.head()
```

item_id	rating		title
	size	sum	
0	452	1756	Toy Story (1995)
1	131	372	GoldenEye (1995)
2	90	267	Get Shorty (1995)
3	192	629	Copycat (1995)
4	59	148	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)

```
popular_movies[['title', 'rating']].head()
```

title	rating	
	size	sum
0	Toy Story (1995)	452 1756
1	GoldenEye (1995)	131 372
2	Get Shorty (1995)	90 267
3	Copycat (1995)	192 629
4	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	59 148

Cosine Similarity

Cosine Similarity measures the similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any angle in the interval (0,π] where the cosine

```
# Import the cosine_similarity function from the sklearn.metrics module
from sklearn.metrics import cosine_similarity
# Create a matrix of user_id and item_id, filling in the rating values
ratings_matrix = ratings_df.pivot_table(index=['user_id'],columns=['item_id'],values=['rating'])
# Fill in NaN values with 0
ratings_matrix = ratings_matrix.fillna(0)
ratings_matrix.head()
```

| | rating | | |