A simple recommendation system for movies

Faculty of Engineering & Technology, Jadavpur University in the partial *fulfillment of the requirements for the degree of Master Of Computer Science And Engineering*

Submitted by

Prosenjit Biswas

Registration Number: 140758 of 2017-18

Class Roll Number: 001710502019

Examination Roll No: *M4CSE19005*

Under the Supervision of

Dr. SarmisthaNeogy

Professor, Dept. of Computer Science & Engineering

Jadavpur University

Dept. of Computer Science & Engineering

Faculty of Engineering and Technology

Jadavpur University

May 2019

Declaration of Originality &

Compliance of Academics Ethics

I hereby declare that this thesis contains literature survey and original research work done by me, as part of my MCSE studies. All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

Name**:** ***Prosenjit Biswas***

Registration No:*140758 of 2017-18*

Class Roll No: *001710502019*

Examination Roll No:*M4CSE19005*

Thesis Report Title:*A simple recommendation system for movies*

_____

ProsenjitBiswas

Department of Computer Science & Engineering

Faculty of Engineering & Technology

Jadavpur University

To Whom It May Concern,

This is to certify thatPROSENJIT BISWAS, Registration Number: 140758 of 2017-18, Class Roll Number: 001710502019, Examination Roll Number: *M4CSE19005*, a student of MCSE, from the Department of Computer Science & Engineering, under the Faculty of Engineering and Technology, Jadavpur University has done a thesis report under my supervision, entitled as "A Simple recommendation system for movies". The thesis is approved for submission towards the fulfillment of the requirements for the degree of Master of *Computer Science &Engineering*, from the Department of Computer Science & Engineering, Jadavpur University for the session 2018-19.

_____

Dr. Sarmistha Neogy
(Supervisor)
Professor
Department of Computer Science and Engineering
Jadavpur University

_____

Dr.Mahantapas Kundu
(Head of the Department)
Professor
Department of Computer Science and Engineering
Jadavpur University

_____

Dr. Chiranjib Bhattacharjee
(Dean)
Professor
Faculty of Engineering and Technology
Jadavpur University

Certificate of Approval

(Only in case the thesis report is approved)

The forgoing thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis only for the purpose for which it is submitted.

_____ _____

Signature of the Examiner Signature of the Examiner

Date: _____ Date: _____

Abstract

Recommendation system plays important role in Internet world and used in many applications. It has created the collection of many application, created global village and growth for numerous information. This report represents the overview of Approaches and techniques generated in recommendation system. Recommendation system is categorized in three classes: Collaborative Filtering, Content based and hybrid based Approach. This report classifies collaborative filtering in two types: Memory based and Model based Recommendation .The paper elaborates these approaches and their techniques with their limitations. The result of our system provides much better recommendations to users because it enables the users to understand the relation between their emotional states and the recommended movies.

Acknowledgement

I would like to express my deepest gratitude to my advisor and guide Dr. SarmisthaNeogy, for her excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I strongly believe that I got a lot of encouragement and inspiration from her throughout the project. With her invaluable guidance, this work is a successful one. I am equally grateful to Dr. MahantapasKundu, Head of The Department, Computer Science & Engineering, Jadavpur University, for his support towards our department. Last but not least, I would like to thank my parents and all respected teachers for their valuable suggestions and helpful discussions.

Regards,

ProsenjitBiswas

Department of Computer Science & Engineering

MCSE

Jadavpur University

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION :

In today's world where internet has become an important part of human life, users often face the problem of too much choice. Right from looking for a motel to looking for good investment options, there is too much information available. To help the users cope with this information explosion, companies have deployed recommendation systems to guide their users. The research in the area of recommendation systems has been going on for several decades now, but the interest still remains high because of the abundance of practical applications and the problem rich domain. A number of such online recommendation systems implemented and used are the recommendation system for books at Amazon.com , for movies at MovieLens.org, CDs at CDNow.com (from Amazon.com), etc.

Recommender Systems have added to the economy of the some of the e-commerce websites (like Amazon.com) and Netflix which have made these systems a salient parts of their websites. A glimpse of the profit of some websites is shown in table below:

Netflix:2/3rd of the movies watched are recommended.
Googles News: recommendation generate 38% more click-throughs.
Amazons:35% sales are from recommendations.
Choicestream: 28% of the people would buy more music if they found what they liked.

Recommender Systems generate recommendations; the user may accept them according to their choice and may also provide, immediately or at a next stage, an implicit or explicit feedback. The actions of the users and their feedbacks can be stored in the recommender database and may be used for generating new recommendations in the next user-system interactions. The economic potential of theses recommender systems have led some of the biggest e-commerce websites (like Amazon.com, snapdeal.com) and the online movie rental company Netflix to make these systems a salient part of their websites. High quality personalized recommendations add another dimension to user experience. The web personalized recommendation systems are recently applied to provide different types of customized information to their respective users. These systems can be applied in various types of applications and are very common now a day.

CHAPTER 2

RELATED WORK

Many recommendation systems have been developed over the past decades. These systems use different approaches like collaborative approach, content based approach, a utility base approach, hybrid approach etc.

Looking at the purchase behavior and history of the shoppers, Lawrence et al. 2001 presented a recommender system which suggests the new product in the market. To refine the recommendation collaborative and content based filtering approach were used. To find the potential customers most of the recommendation systems today use ratings given by previous users. These ratings are further used to predict and recommend the item of one's choice.

In 2007 Weng, Lin and Chen performed an evaluation study which says using multidimensional analysis and additional customer's profile increases the recommendation quality. Weng used MD recommendation model (multidimensional recommendation model) for this purpose. multidimensional recommendation model was proposed by Tuzhilin and Adomavicius (2001).

CHAPTER 3

PHASES OF RECOMMENDATION PROCESS:

The system normally asks a person via the device interface to provide rating for items to construct and enhance his version. The accuracy of recommendation relies upon on the amount of ratings provided via the person. The only shortcoming of this method is, it requires effort from the customers and additionally, customers may not be always ready to provide sufficient facts. Despite the fact that specific feedback needs more effort from consumer, it is still seen as presenting extra dependable information, since it does now not involve extracting options from moves, and it additionally provides transparency into the recommendation system that effects in a slightly better perceived recommendation great and greater confidence inside the recommendations

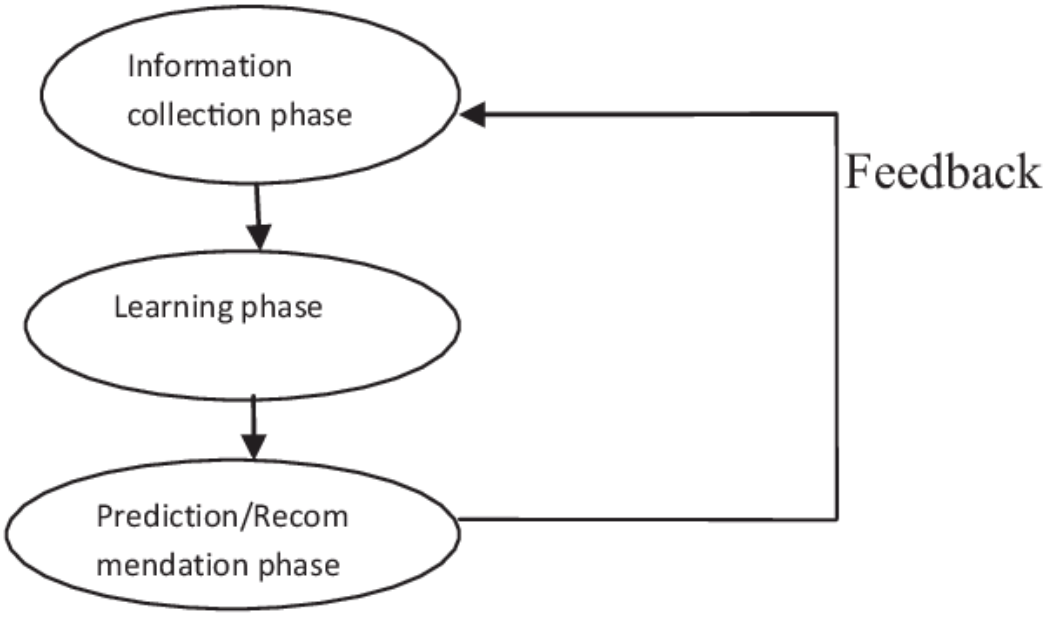


Fig1:Recommendation Phases

The use of efficient and accurate recommendation techniques is very important for a system that will provide good and useful recommendation to its individual users. This explains the importance of understanding the features and potentials of different recommendation techniques.

CHAPTER 4

RECOMMENDER SYSTEMBASICS CONCEPT

Recommender systems are information filtering tools that are used to predict the rating for users and items, predominantly from big data to recommend their likes. Movie recommendation systems provide a mechanism to assist users in classifying users with similar interests. This makes recommender systems essentially a central part of websites and e-commerce applications.

In this project we propose a movie recommendation system, where user specific interests are taken into account, to determine recommendations.

A Recommendation System is composed of two modules: a database and a filtering technique. The database is responsible for storing the information about users, items and the associated ratings. The filtering technique is implemented by an algorithm.There are three important types of recommender systems:

- Collaborative Filtering
- Content based Filtering
- Hybrid Filtering

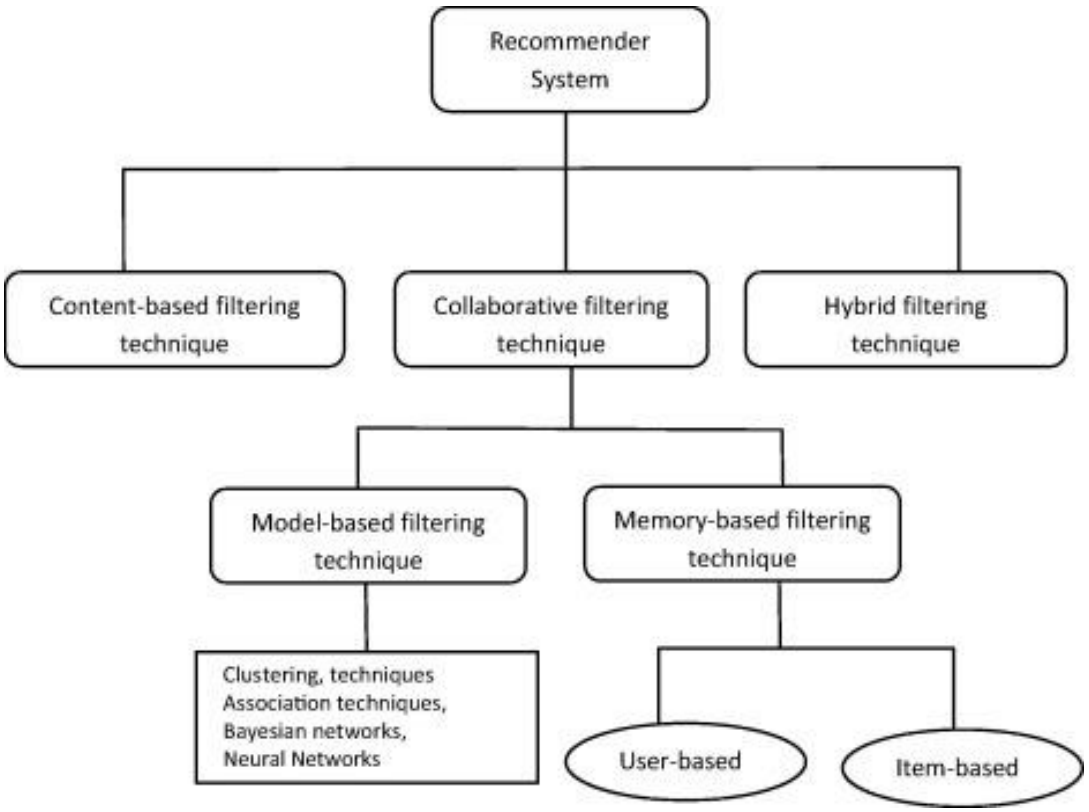


Fig2:Recommender system based on filtering

4.1.Content-Based Filtering:

Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions. When documents such as web pages, publications and news are to be recommended, content-based filtering technique is the most successful. In content-based filtering technique, recommendation is made based on the user profiles using features extracted from the content of the items the user has evaluated in the past. Items that are mostly related to the positively rated items are recommended to the user. CBF uses different types of models to find similarity between documents in order to generate meaningful recommendations. It could use Vector Space Model such as Term Frequency Inverse Document Frequency (TF/IDF) or Probabilistic models such as Decision Trees. Neural Networks to model the relationship between different documents within a corpus. These techniques make recommendations by learning the underlying model with either statistical analysis or machine learning techniques. Content-based filtering technique does not need the profile of other users since they do not influence recommendation. Also, if the user profile changes, CBF technique still has the potential to adjust its recommendations within a very short period of time. The major disadvantage of this technique is the need to have an in-depth knowledge and description of the features of the items in the profile.

4.2Collaborative Filtering:

Collaborative filtering is a domain-independent prediction technique for content that cannot easily and adequately be described by metadata such as movies and music. Collaborative filtering technique works by building a database (user-item matrix) of preferences for items by users. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations. Such users build a group called neighborhood. A user gets recommendations to those items that he has not rated before but that were already positively rated by users in his neighborhood. Recommendations that are produced by CF can be of either prediction or recommendation. Prediction is a numerical value, Rij, expressing the predicted score of item j for the user i, while Recommendation is a list of top N items that the user will like the most. The technique of collaborative filtering can be divided into two categories: memory-based and model-based .

4.2.1.Model-Based Techniques:

This technique employs the previous ratings to learn a model in order to improve the performance of Collaborative filtering Technique. The model building process can be done using machine learning or data mining techniques. These techniques can quickly recommend a set of items for the fact that they use pre-computed model and they have proved to produce recommendation results that are similar to neighborhood-based recommender techniques. Examples of these techniques include Dimensionality Reduction technique such as Singular Value Decomposition (SVD), Matrix Completion Technique, Latent Semantic methods, and Regression and Clustering. Model-based techniques analyze the user-item matrix to identify relations between items; they use these relations to compare the list of top-N recommendations. Model based techniques resolve the sparsity problems associated with recommendation systems.

Model-based Collaborative Filtering is based on matrix factorization(MF)which has received greater exposure, mainly as an unsupervised learning method for latent variable decomposition and dimensionality reduction. Matrix factorization is widely used for recommender systems where it can deal better with scalability and sparsity than Memory-based Collaborative Filtering. The goal of MF is to learn the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and items. When you have a very sparse matrix, with a lot of dimensions, by doing matrix factorization you can restructure the user-item matrix into low-rank structure, and you can represent the matrix by the multiplication of two low-rank matrices, where the rows contain the latent vector. You fit this matrix to approximate your original matrix, as closely as possible, by multiplying the low-rank matrices together, which fills in the entries missing in the original matrix .

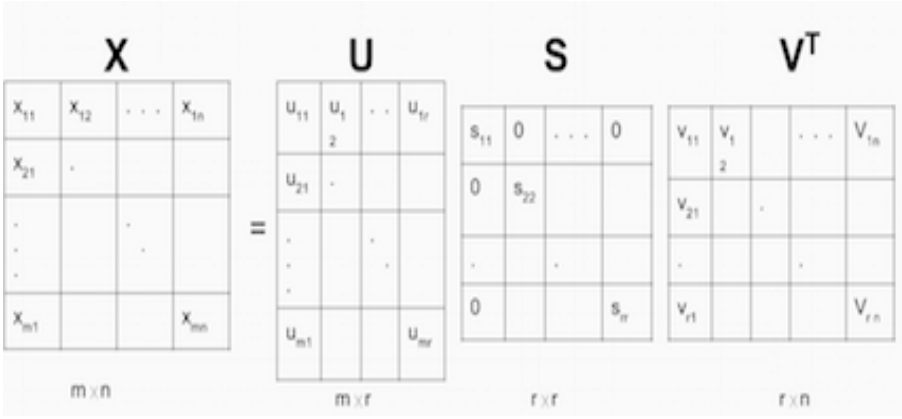
A well-known matrix factorization method is Singular value decomposition (SVD). Collaborative Filtering can be formulated by approximating a matrix **X** by using singular value decomposition.The general equation can be expressed as follows:

$$\mathbf{X}=\mathbf{U}\times\mathbf{S}\times\mathbf{V}\mathbf{T}$$

Given an$\mathbf{m}\times\mathbf{n}$ matrix$\mathbf{X}$:

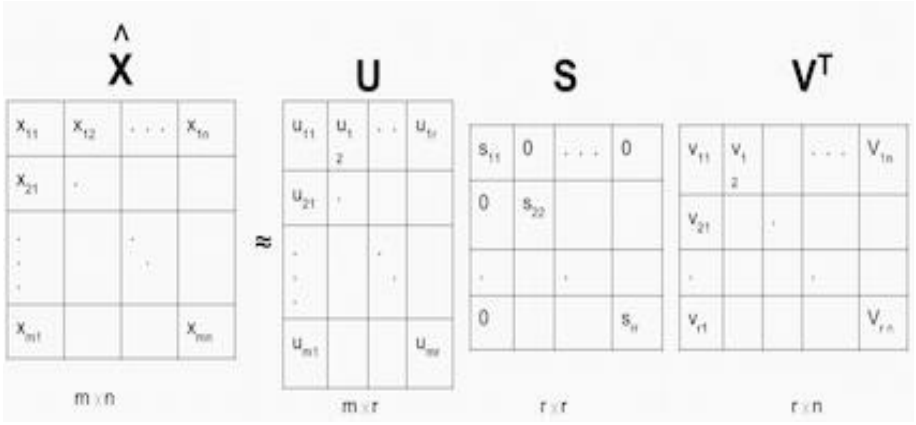
- **U** is an **m** x **r** orthogonal matrix
- **S** is an **r** x **r** diagonal matrix with non-negative real numbers on the diagonal
- **VT** is an **r** x **n** orthogonal matrix

Elements on the diagnoal in **S** are known as singular values of **X**.
Matrix **X** can be factorized to **U**, **S** and **V**. The **U** matrix represents the feature vectors corresponding to the users in the hidden feature space and the **V** matrix represents the feature vectors corresponding to the items in the hidden feature space.



https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

Now we can make a prediction by taking dot product of **U**, **S** and **VT**.



https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

Just as its name suggest matrix factorization is to, obviously, factorize a matrix, i.e. to find out two (or more) matrices such that when you multiply them you will get back the original matrix. matrix factorization can be used to discover latent features underlying the interactions between two different kinds of entities. (Of course, you can consider more than two kinds of entities and you will be dealing with tensor factorization, which would be more complicated.) And one obvious application is to predict ratings in collaborative filtering.

In a recommendation system such as Netflix or MovieLens , there is a group of users and a set of items (movies for the above two systems). Given that each users have rated some items

in the system, we would like to predict how the users would rate the items that they have not yet rated, such that we can make recommendations to the users. In this case, all the information we have about the existing ratings can be represented in a matrix. Assume now we have 5 users and 10 items, and ratings are integers ranging from 1 to 5, the matrix may look something like this (a hyphen means that the user has not yet rated the movie):

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

Hence, the task of predicting the missing ratings can be considered as filling in the blanks (the hyphens in the matrix) such that the values would be consistent with the existing ratings in the matrix.

The intuition behind using matrix factorization to solve this problem is that there should be some latent features that determine how a user rates an item. For example, two users would give high ratings to a certain movie if they both like the actors/actresses of the movie, or if the movie is an action movie, which is a genre preferred by both users. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item.

In trying to discover the different features, we also make the assumption that the number of features would be smaller than the number of users and the number of items. It should not be difficult to understand this assumption because clearly it would not be reasonable to assume that each user is associated with a unique feature (although this is not impossible). And anyway if this is the case there would be no point in making recommendations, because each of these users would not be interested in the items rated by other users. Similarly, the same argument applies to the items.

➢ ***MATHEMATICS OF MATRIX FACTORIZATION*:**

Having discussed the intuition behind matrix factorization, we can now go on to work on the mathematics. Firstly, we have a set U of users, and a set D of items. Let **R** of size **|U|x|D|** be the matrix that contains all the ratings that the users have assigned to the items.

Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrices **P** (a **|U|xK** matrix) and **Q** (a **|D|xK** matrix) such that their product approximates **R**

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

In this way, each row of $\mathbf{P}$ would represent the strength of the associations between a user and the features. Similarly, each row of $\mathbf{Q}$ would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item $\mathbf{d_j}$ by $\mathbf{U_i}$, we can calculate the dot product of the two vectors corresponding to $\mathbf{U_i}$ and $\mathbf{d_j}$:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

Now, we have to find a way to obtain $\mathbf{P}$ and $\mathbf{Q}$. One way to approach this problem is the first initialize the two matrices with some values, calculate how `different' their product is to $\mathbf{M}$, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference .

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

Here we consider the squared error because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of $\mathbf{p_{ik}}$ and $\mathbf{q_{kj}}$. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik}$$

Having obtained the gradient, we can now formulate the update rules for both $\mathbf{p_{ik}}$ and $\mathbf{q_{kj}}$:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

Here, $\boldsymbol{\alpha}$ is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for $\boldsymbol{\alpha}$, say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

we are not really trying to come up with **P** and **Q** such that we can reproduce **R** exactly. Instead, we will only try to minimise the errors of the observed user-item pairs. In other words, if we let **T** be a set of tuples, each of which is in the form of $\mathbf{(u_i,d_j,r_{ij})}$, such that T contains all the observed user-item pairs together with the associated ratings, we are only trying to minimise every $\mathbf{e_{ij}}$ for $\mathbf{(u_i,d_j,r_{ij})\varepsilon T}$. (In other words, **T** is our set of training data.) As for the rest of the

unknowns, we will be able to determine their values once the associations between the users, items and features have been learnt.
Using the above update rules, we can then iteratively perform the operation until the error converges to its minimum. We can check the overall error as calculated using the following equation and determine when we should stop the process .

$$E = \sum_{(u_i,d_j,r_{ij})\in T} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T} (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})^2$$

- ***<u>REGULARIZATION</u>*:**

The above algorithm is a very basic algorithm for factorizing a matrix. There are a lot of methods to make things look more complicated. A common extension to this basic algorithm is to introduce regularization to avoid overfitting. This is done by adding a parameter **β** and modify the squared error as follows:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})^2 + \frac{\beta}{2}\sum_{k=1}^{K}(||P||^2 + ||Q||^2)$$

In other words, the new parameter **β** is used to control the magnitudes of the user-feature and item-feature vectors such that **P** and **Q** would give a good approximation of R without having to contain large numbers. In practice, **β** is set to some values in the range of 0.02. The new update rules for this squared error can be obtained by a procedure similar to the one described above. The new update rules are as follows.

$$p'_{ik} = p_{ik} + \alpha\frac{\partial}{\partial p_{ik}}e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik})$$
$$q'_{kj} = q_{kj} + \alpha\frac{\partial}{\partial q_{kj}}e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj})$$

<u>4.2.2.Memory Based Techniques</u> :

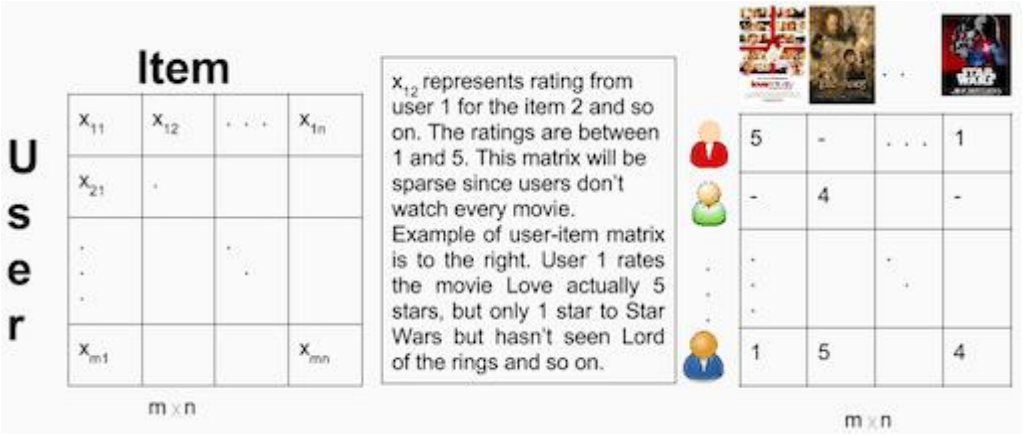
The items that were already rated by the user before play a relevant role in searching for a neighbor that shares appreciation with him .Once a neighbor of a user is found, different algorithms can be used to combine the preferences of neighbors to generate recommendations. Due to the effectiveness of these techniques, they have achieved widespread success in real life applications. Memory-based CF can be achieved in two ways through user-based and item-based techniques. User based collaborative filtering technique calculates similarity between users by comparing their ratings on the same item, and it then computes the predicted rating for an item by the active user as a weighted average of the ratings of the item by users similar to the active user where weights are the similarities of these users with the target item. Item-based filtering techniques compute predictions using the similarity between items and not the similarity between users. It builds a model of item similarities by retrieving all items rated by an active user from the user-item matrix, it determines how similar the retrieved items are to the target item, then it

selects the k most similar items and their corresponding similarities are also determined. Prediction is made by taking a weighted average of the active users rating on the similar items k.

Memory-based algorithms are easy to implement and produce reasonable prediction quality.This Collaborative Filtering approaches can be divided into two main sections: user-item filtering and item-item filtering. A user-item filtering takes a particular user, find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked. In contrast, item-item filtering will take an item, find users who liked that item, and find other items that those users or similar users also liked. It takes items and outputs other items as recommendations .

- Item-Item Collaborative Filtering: “Users who liked this item also liked …”
- User-Item Collaborative Filtering: “Users who are similar to you also liked …”

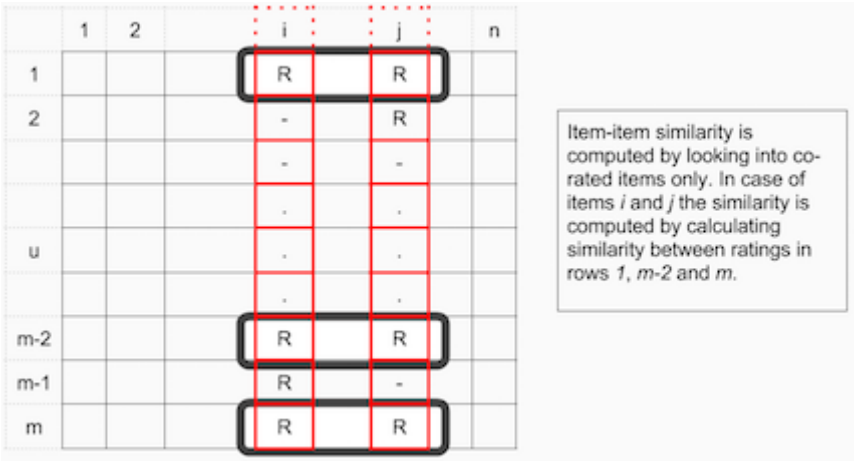
In both cases, we create a user-item matrix whichwe build from the entire dataset. Since we have split the data into testing and training we will need to create two 943 ×× 1682 matrices. The training matrix contains 75% of the ratings and the testing matrix contains 25% of the ratings.
Example of user-item matrix:



https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

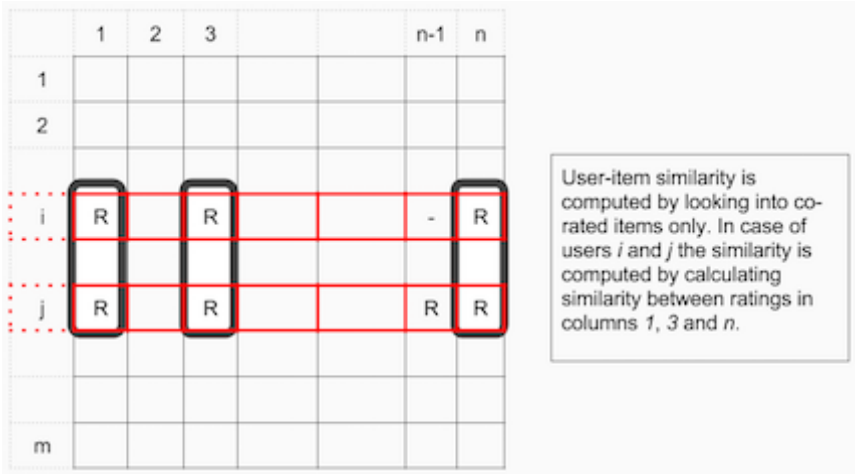
After building the user-item matrix we calculate the similarity and create a similarity matrix.

The similarity values between items in Item-Item Collaborative Filtering are measured by observing all the users who have rated both items.



https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

For User-Item Collaborative Filtering the similarity values between users are measured by observing all the items that are rated by both users.



https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

A distance metric commonly used in recommender systems is cosine similarity, where the ratings are seen as vectors in n-dimensional space and the similarity is calculated based on the angle between these vectors. Cosine similarity for users a and m can be calculated using the formula below, where you take dot product of the user vector U_k and the user vectorU_a and divide it by multiplication of the Euclidean lengths of the vectors.

$$s_u^{cos}(u_k, u_a) = \frac{u_k \cdot u_a}{\|u_k\| \, \|u_a\|} = \frac{\sum x_{k,m} x_{a,m}}{\sqrt{\sum x_{k,m}^2 \sum x_{a,m}^2}}$$

To calculate similarity between items ***m*** and ***b*** you use the formula:

$$s_u^{cos}(i_m, i_b) = \frac{i_m \cdot i_b}{\|i_m\| \, \|i_b\|} = \frac{\sum x_{a,m} x_{a,b}}{\sqrt{\sum x_{a,m}^2 \sum x_{a,b}^2}}$$

Now we can make a prediction by applying following formula for user-based Collaborative Filtering:

$$\hat{x}_{k,m} = \bar{x}_k + \frac{\sum\limits_{u_a} sim_u(u_k, u_a)(x_{a,m} - \bar{x}_{u_a})}{\sum\limits_{u_a} |sim_u(u_k, u_a)|}$$

We can look at the similarity between users ***k*** and ***a*** as weights that are multiplied by the ratings of a similar user ***a*** (corrected for the average rating of that user). We need to normalize it so that the ratings stay between 1 and 5 and, as a final step, sum the average ratings for the user that we are trying to predict .

The idea here is that some users may tend always to give high or low ratings to all movies. The relative difference in the ratings that these users give is more important than the absolute values. To give an example: suppose, user ***k*** gives 4 stars to his favourite movies and 3 stars to all other good movies. Suppose now that another user ***t*** rates movies that he/she likes with 5 stars, and the movies he/she fell asleep over with 3 stars. These two users could have a very similar taste but treat the rating system differently.

When making a prediction for item-based CF we don't need to correct for users average rating since query user itself is used to do predictions.

$$\hat{x}_{k,m} = \frac{\sum\limits_{i_b} sim_i(i_m, i_b)(x_{k,b})}{\sum\limits_{i_b} |sim_i(i_m, i_b)|}$$

The most popular metric used to evaluate accuracy of predicted ratings is *Root Mean Squared Error (RMSE).*

$$RMSE = \sqrt{\frac{1}{N} \sum (x_i - \hat{x}_i)^2}$$

- ***Problems of Collaborative Filtering Techniques***:

Collaborative Filtering has some major advantages over CBF in that it can perform in domains where there is not much con-tent associated with items and where content is difficult for a computer system to analyze (such as opinions and ideal). Also, CF technique has the ability to provide serendipitous recommendations, which means that it can recommend items that are relevant to the user even without the content being in the user's profile . Despite the success of CF techniques, their widespread use has revealed some potential problems such as follows

- ***Cold-Start Problem:***

This refers to a situation where a recommender does not have adequate information about a user or an item in order to make relevant predictions. This is one of the major problems that reduce the performance of recommendation system. The profile of such new user or item will be empty since he has not rated any item; hence, his taste is not known to the system .

- ***Data Sparsity Problem:***

This is the problem that is occur as a result of lack of enough information, that is, when only a few of the total number of items available in a database are rated by users . This always leads to a sparse user-item matrix, inability to locate successful neighbors and finally, the generation of weak recommendations. Also, data sparsity always leads to coverage problems, which is the percentage of items in the system that recommendations can be made for.

- ***Scalability:***

This is another problem associated with Recommendation algorithms because computation normally grows linearly with the number of users and items .A recommendation technique that is efficient when the number of dataset is limited may be unable to generate satisfactory number of recommendations when the volume of dataset is increased. Thus, it is crucial to apply recommendation techniques which are capable of scaling up in a successful manner as the number of dataset in a database increases. Methods used for solving scalability problem and speeding up recommendation generation are based on Dimensionality reduction techniques, such as Singular Value Decomposition (SVD) method, which has the ability to produce reliable and efficient recommendations.

- ***Examples of Collaborative Systems****:*

Ringo is a user-based CF system which makes recommendations of music albums and artists. In Ringo, when a user initially enters the system, a list of 125 artists is given to the user to rate according to how much he likes listening to them. The list is made up of two different sections. The first session consists of the most often rated artists, and this affords the active user opportunity to rate artists which others have equally rated, so that there is a level of similarities between different users' profiles. The second session is generated upon a random selection of items from the entire user-item matrix, so that all artists and albums are eventually rated at some point in the initial rating phases.

Group Lens is a CF system that is based on client/server architecture; the system recommends Usenet news which is a high volume discussion list service on the Internet. The short lifetime of Netnews, and the underlying sparsity of the rating matrices are the two main challenges addressed by this system. Users and Netnews are clustered based on the existing news groups in the system, and the implicit ratings are computed by measuring the time the users spend reading Netnews.

Amazon.com is an example of e-commerce recommendation engine that uses scalable item-to-item collaborative filtering techniques to recommend online products for different users. The computational algorithm scales independently of the number of users and items within the database. Amazon.com uses an explicit information collection technique to obtain information from users. The interface is made up of the following sections, your browsing history, rate these items, and improve your recommendations and your profile. The sys-tem predicts users interest based on the items he/she has rated

4.3.Hybrid Filtering:

Hybrid filtering technique combines different recommendation techniques in order to gain better system optimization to avoid some limitations and problems of pure recommendation systems. The idea behind hybrid techniques is that a combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm .Using multiple recommendation techniques can suppress the weaknesses of an individual technique in a combined model. The combination of approaches can be done in any of the following ways: separate implementation of algorithms and combining the result, utilizing some content-based filtering in collaborative approach, utilizing some collaborative filtering in content-based approach, creating a unified recommendation system that brings together both approaches.

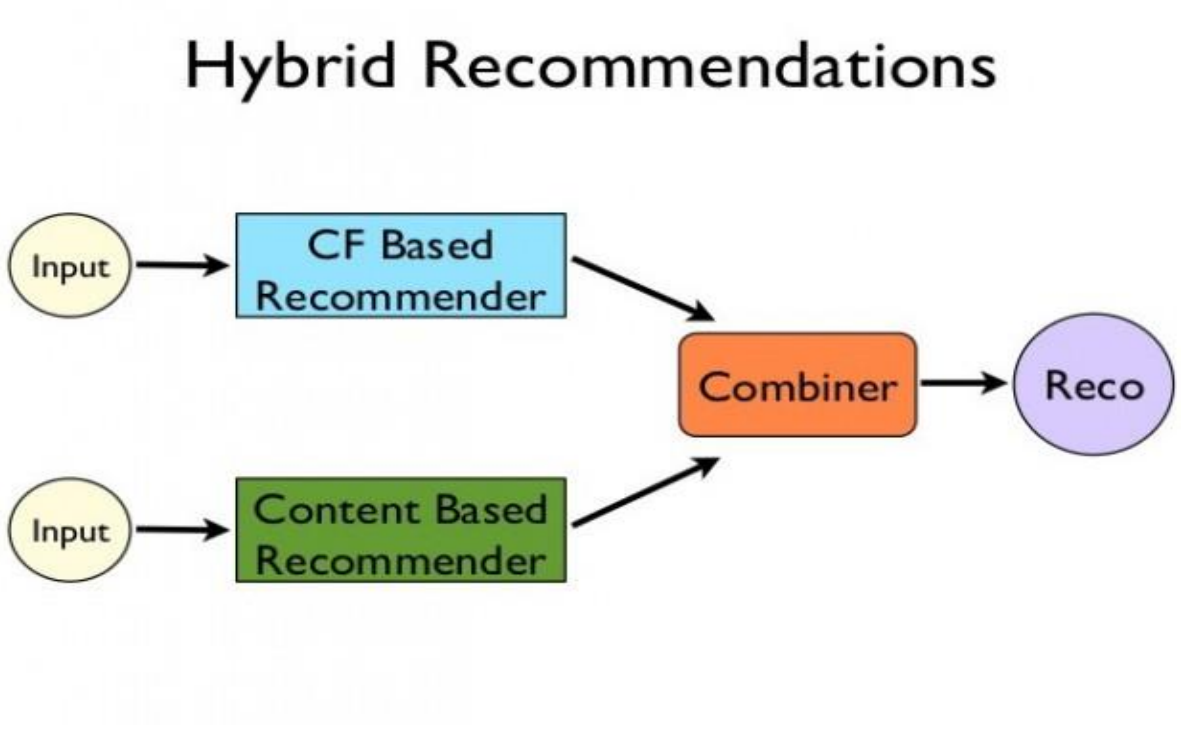


Fig3: Hybrid system

CHAPTER 5

➢ DESIGN AND IMPLENTATION OF A RECOMMENDER SYSTEM FOR MOVIE:

5.1DATA SET:

We will be using the MovieLensdataset forrecommendation. It has been collected by the Group Lens research project at the University of Minnesota/. The characteristics of the MovieLens ml-100K datasetare as follows:

- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.
- Simple demographic info for the uses(age, gender, occupation.zip)
- Genre information of movies.

This data is loaded into python. There are many files in ml-100k.zip file which we used. We loaded three important files. There is also a recommendation to read the readme document which gives a lot of information about difference files.

5.1.1.USERS :

There are 943 rows as there are 943 users and 5 columns for 5 featured for each namely their unique user_id, age, sex, occupation, zip_code.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 943 entries, 0 to 942
Data columns (total 5 columns):
user id       943 non-null int64
age           943 non-null int64
gender        943 non-null object
```

occupation 943 non-null object

zip code 943 non-null object

dtypes: int64(2), object(3)

memory usage: 25.8+ KB

None

We can print the head of the user dataset as follows:

	user id	age	gender	occupation	zip code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
7	8	36	M	administrator	05201
9	9	29	M	student	01002
9	10	53	M	lawyer	90703

5.1.2.RATINGS :

There are 100000 ratings as there are 100K ratings for different users and movie combinations. Also each ratings has a timestamp associated with it.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 100000 entries, 0 to 99999

Data columns (total 4 columns):

user id 100000 non-null int64

movie id 100000 non-null int64

rating 100000 non-null int64

timestamp 100000 non-null int64

dtypes: int64(4)

memory usage: 3.1 MB

None

We can print the head of the data dataset as follows:

	user id	movie 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
5	298	474	4	884182806
6	115	265	2	881171488
7	253	465	5	891628467
8	305	451	3	886324817
9	6	86	3	883603013

5.1.3.ITEMS :

This dataset contains attributes of 1682 movies. There are 24 columns out of which 19 specify the genre of a particular movie. The last 19 columns are for each genre and a value of 1 denotes that movie belongs to that genre and 0 otherwise.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1682 entries, 1 to 1682
Data columns (total 24 columns):
movie id              1682 non-null object
movie title           1681 non-null object
release date          0 non-null float64
video release date    1679 non-null object
IMDb URL              1682 non-null int64
unknown               1682 non-null int64
Action                1682 non-null int64
Adventure             1682 non-null int64
Animation             1682 non-null int64
Childrens             1682 non-null int64
Comedy                1682 non-null int64
Crime                 1682 non-null int64
Documentary           1682 non-null int64
Drama                 1682 non-null int64
Fantasy               1682 non-null int64
Film-Noir             1682 non-null int64
Horror                1682 non-null int64
Musical               1682 non-null int64
Mystery               1682 non-null int64
Romance               1682 non-null int64
Sci-Fi                1682 non-null int64
Thriller              1682 non-null int64
War                   1682 non-null int64
```

```
Western         1682 non-null object
dtypes: float64(1), int64(20), object(3)
memory usage: 295.7+ KB
None
```

We can print the head of the item dataset as follows:

```
   movie id  ...  Western
0         1  ...        0
1         2  ...        0
2         3  ...        0
3         4  ...        0
4         5  ...        0
5         6  ...        0
6         7  ...        0
7         8  ...        0
8         9  ...        0
9        10  ...        0

[10 rows x 24 columns]
```

5.2.IMPLEMENTATION:

We have implemented movie recommendation system in python .we have used various library functions. We have used panda We used the scikit-learn library to split the dataset into testing and training.

- We have used **panda** which is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

- We have used **NumPy** is the fundamental package for scientific computing with Python.

- Also we have used **codecs** which defines a set of base classes which define the interface and can also be used to easily write your own **codecs** for use in Python.

- We have used standard **operators** as functions. For example, **operator.add(x, y)** is equivalent to the expression x+y. Many function names are those used for special methods, without the double underscores. For backward compatibility, many of these have a variant with the double underscores kept. The variants without the double underscores are preferred for clarity.

- We have also used **importlib** thepurpose this package is two-fold. One is to provide the implementation of the import statement (and thus, by extension, the __import__() function) in Python source code. This provides an implementation of **import** which is portable to any Python interpreter. This also provides an implementation which is easier to comprehend than one implemented in a programming language other than Python. Two, the components to implement **import** are exposed in this package, making it easier for users to create their own custom objects (known generically as an importer) to participate in the import process.

- We have also imported **Scripy.parse** which is SciPy 2-D sparse matrix package for numeric data.

5.3WORK FLOW AND RESULTS:

5.3.1.SIMPLE RECOMMENDATION: It's a generalized recommendation for every user , based on movie popularity. More popular and critically acclaimed will have a higher probability of being liked by the average audience. IMDB Top 250 is an example of this system.

The dataset has 3 comma separated files namely u.data, u.item, u.user.

- 25 MOST RATED MOVIES:

WORKFLOW:

I. Merge the 3 dataset user, data and item.

II. Name the merged data set as “dataset”.

III. Count the number of times a movie is rated by user which is group by “movie title” .

IV. Make the total ratings into a dataframe and sort them into the descending order and print the 25top rated movies.

OUTPUT:

	movie id	movie title	...	occupation	zip code
0	1	Toy Story (1995)	...	retired	95076
1	4	Get Shorty (1995)	...	retired	95076
2	5	Copycat (1995)	...	retired	95076
3	7	Twelve Monkeys (1995)	...	retired	95076
4	8	Babe (1995)	...	retired	95076
5	9	Dead Man Walking (1995)	...	retired	95076
6	11	Seven (Se7en) (1995)	...	retired	95076
7	12	Usual Suspects, The (1995)	...	retired	95076

[8 rows x 31 columns]

15 random order rated movies:

0	'Til There Was You (1997)	9
1	1-900 (1994)	5
2	101 Dalmatians (1996)	109
3	12 Angry Men (1957)	125
4	187 (1997)	41
5	2 Days in the Valley (1996)	93
6	20,000 Leagues Under the Sea (1954)	72
7	2001: A Space Odyssey (1968)	259
8	3 Ninjas: High Noon At Mega Mountain (1998)	5

9	39 Steps, The (1935)	59
10	8 1/2 (1963)	38
11	8 Heads in a Duffel Bag (1997)	4
12	8 Seconds (1994)	4
13	A Chef in Love (1996)	8
14	Above the Rim (1994)	5

25 Top rated movies:

	movie title	total ratings
1398	Star Wars (1977)	583
333	Contact (1997)	509
498	Fargo (1996)	508
1234	Return of the Jedi (1983)	507
860	Liar Liar (1997)	485
460	English Patient, The (1996)	481
1284	Scream (1996)	478
1523	Toy Story (1995)	452
32	Air Force One (1997)	431
744	Independence Day (ID4) (1996)	429
1205	Raiders of the Lost Ark (1981)	420
612	Godfather, The (1972)	413
1190	Pulp Fiction (1994)	394
1543	Twelve Monkeys (1995)	392
1329	Silence of the Lambs, The (1991)	390
780	Jerry Maguire (1996)	384
293	Chasing Amy (1997)	379
1251	Rock, The (1996)	378
456	Empire Strikes Back, The (1980)	367
1394	Star Trek: First Contact (1996)	365
1500	Titanic (1997)	350
113	Back to the Future (1985)	350
987	Mission: Impossible (1996)	344
570	Fugitive, The (1993)	336
747	Indiana Jones and the Last Crusade (1989)	331

- ***Highly rated movies***:

I. Group the movies by movie title from the dataset "dataset".
II. Evaluating the Mean[mean is the average of list of values] of each movies.
III. Store the mean rated movies into dataframes " ratings_mean"
IV. Sort them by descending order and print the 25 highly rated movies,

Mean rating of movies in random order:

	mean ratings	movie title
0	2.333333	'Til There Was You (1997)
1	2.600000	1-900 (1994)
2	2.908257	101 Dalmatians (1996)
3	4.344000	12 Angry Men (1957)
4	3.024390	187 (1997)
5	3.225806	2 Days in the Valley (1996)
6	3.500000	20,000 Leagues Under the Sea (1954)
7	3.969112	2001: A Space Odyssey (1968)
8	1.000000	3 Ninjas: High Noon At Mega Mountain (1998)
9	4.050847	39 Steps, The (1935)
10	3.815789	8 1/2 (1963)
11	3.250000	8 Heads in a Duffel Bag (1997)
12	3.750000	8 Seconds (1994)
13	4.125000	A Chef in Love (1996)
14	3.000000	Above the Rim (1994)

25 highly rated movies:

	mean ratings	movie title
1472	5.000000	They Made Me a Criminal (1939)
944	5.000000	Marlene Dietrich: Shadow and Light (1996)
1273	5.000000	Saint of Fort Washington, The (1993)
1359	5.000000	Someone Else's America (1995)
1387	5.000000	Star Kid (1997)
633	5.000000	Great Day in Harlem, A (1994)
30	5.000000	Aiqingwansui (1994)
1277	5.000000	Santa with Muscles (1996)
1172	5.000000	Prefontaine (1997)
462	5.000000	Entertaining Angels: The Dorothy Day Story (1996)
1130	4.625000	PatherPanchali (1955)
1357	4.500000	Some Mother's Son (1996)
956	4.500000	Maya Lin: A Strong Clear Vision (1994)
79	4.500000	Anna (1996)
472	4.500000	Everest (1998)
318	4.491071	Close Shave, A (1995)
1281	4.466443	Schindler's List (1993)
1652	4.466102	Wrong Trousers, The (1993)
273	4.456790	Casablanca (1942)
1597	4.447761	Wallace &Gromit: The Best of AardmanAnimatio...
1317	4.445230	Shawshank Redemption, The (1994)
1215	4.387560	Rear Window (1954)

1572	4.385768	Usual Suspects, The (1995)
1398	4.358491	Star Wars (1977)
3	4.344000	12 Angry Men (1957)

- **Mean rating of movie along with no of ratings of user :**

I. Evaluate the mean rating of each movie then count the number of user who gives rating and group meand ratings and total ratings by "movie title"
II. Print 25 mivies in random order.
III. Sort the movies on the basic of total rating and print the top 24 movies along with mean rating and total rating(descending order).
IV. print the top 10 movies along with mean rating and total rating(descending order) where total ratings less than 300.

Random order:

movie title	Rating	no of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41
2 Days in the Valley (1996)	3.225806	93
20,000 Leagues Under the Sea (1954)	3.500000	72
2001: A Space Odyssey (1968)	3.969112	259
3 Ninjas: High Noon At Mega Mountain (1998)	1.000000	5
39 Steps, The (1935)	4.050847	59
8 1/2 (1963)	3.815789	38
8 Heads in a Duffel Bag (1997)	3.250000	4
8 Seconds (1994)	3.750000	4
A Chef in Love (1996)	4.125000	8
Above the Rim (1994)	3.000000	5
Absolute Power (1997)	3.370079	127
Abyss, The (1989)	3.589404	151
Ace Ventura: Pet Detective (1994)	3.048544	103
Ace Ventura: When Nature Calls (1995)	2.675676	37
Across the Sea of Time (1995)	2.750000	4
Addams Family Values (1993)	2.816092	87
Addicted to Love (1997)	3.166667	54
Addiction, The (1995)	2.181818	11
Adventures of Pinocchio, The (1996)	3.051282	39
Adventures of Priscilla, Queen of the Desert, T...	3.594595	111

25movies in sorted order:

	rating	movie title	total ratings
1398	4.358491	Star Wars (1977)	583
333	3.803536	Contact (1997)	509
498	4.155512	Fargo (1996)	508
1234	4.007890	Return of the Jedi (1983)	507
860	3.156701	Liar Liar (1997)	485
460	3.656965	English Patient, The (1996)	481
1284	3.441423	Scream (1996)	478
1523	3.878319	Toy Story (1995)	452
32	3.631090	Air Force One (1997)	431
744	3.438228	Independence Day (ID4) (1996)	429
1205	4.252381	Raiders of the Lost Ark (1981)	420
612	4.283293	Godfather, The (1972)	413
1190	4.060914	Pulp Fiction (1994)	394
1543	3.798469	Twelve Monkeys (1995)	392
1329	4.289744	Silence of the Lambs, The (1991)	390
780	3.710938	Jerry Maguire (1996)	384
293	3.839050	Chasing Amy (1997)	379
1251	3.693122	Rock, The (1996)	378
456	4.204360	Empire Strikes Back, The (1980)	367
1394	3.660274	Star Trek: First Contact (1996)	365
1500	4.245714	Titanic (1997)	350
113	3.834286	Back to the Future (1985)	350
987	3.313953	Mission: Impossible (1996)	344
570	4.044643	Fugitive, The (1993)	336

- **Top 10 highly rated under total ratings 300**:

	rating	movie title	total ratings
1281	4.466443	Schindler's List (1993)	298
1652	4.466102	Wrong Trousers, The (1993)	118
273	4.456790	Casablanca (1942)	243
1317	4.445230	Shawshank Redemption, The (1994)	283
1215	4.387560	Rear Window (1954)	209
1572	4.385768	Usual Suspects, The (1995)	267
1398	4.358491	Star Wars (1977)	583
3	4.344000	12 Angry Men (1957)	125
303	4.292929	Citizen Kane (1941)	198
1507	4.292237	To Kill a Mockingbird (1962)	219

- ***Mean rating of movies on genderwise***:

I. From the dataset, we find the mean rating of movies according to the given rating of male and female. Then print the 25 movies in random order.

II. We find the movies whose total ratings are greater than or equal to300. Then among these movies, we find the mean rating of movies according to the given rating of male and female. Then print the 15movies in random order.

III. Then we find the highly rated movies based on female rating and then print 15 highly rated movies on female rating.

IV. We find the difference between mean ratings given by male and female.Print 15 movies.

V. Sort the movies in descending order based on the difference and then print the 15 movies which are most disagree on male and female. (Here all the movies whose total ratings are greater than or equal to 300).

- ***Mean rating of movies based on male and female***:

	gender	
	F	M
movie title		
'Til There Was You (1997)	2.200000	2.500000
1-900 (1994)	1.000000	3.000000
101 Dalmatians (1996)	3.116279	2.772727
12 Angry Men (1957)	4.269231	4.363636
187 (1997)	3.500000	2.870968
2 Days in the Valley (1996)	3.235294	3.223684

20,000 Leagues Under the Sea (1954) 3.214286 3.568966

2001: A Space Odyssey (1968) 3.491228 4.103960

3 Ninjas: High Noon At Mega Mountain (1998 1.00000 1.000000

39 Steps, The (1935) 4.000000 4.060000

8 1/2 (1963) 2.800000 3.969697

8 Heads in a Duffel Bag (1997) 2.500000 4.000000

8 Seconds (1994) 4.000000 3.666667

A Chef in Love (1996) 3.750000 4.500000

Above the Rim (1994) NaN 3.000000

Absolute Power (1997) 3.451613 3.343750

Abyss, The (1989) 3.814815 3.540323

Ace Ventura: Pet Detective (1994) 3.105263 3.035714

Ace Ventura: When Nature Calls (1995) 2.300000 2.814815

Across the Sea of Time (1995) 2.666667 3.000000

Addams Family Values (1993) 3.000000 2.757576

Addicted to Love (1997) 3.074074 3.259259

Addiction, The (1995) 2.000000 2.222222

Adventures of Pinocchio, The (1996) 2.909091 3.107143

Adventures of Priscilla, Queen of the Desert, T... 3.659091 3.552239

- ***Movies whose rating are greater than or equal to 300*:**

Index([u'Air Force One (1997)', u'Back to the Future (1985)',

u'Chasing Amy (1997)', u'Contact (1997)',

u'E.T. the Extra-Terrestrial (1982)',

u'Empire Strikes Back, The (1980)', u'English Patient, The (1996)',

u'Fargo (1996)', u'Forrest Gump (1994)', u'Fugitive, The (1993)',

u'Full Monty, The (1997)', u'Godfather, The (1972)',

u'Independence Day (ID4) (1996)',

u'Indiana Jones and the Last Crusade (1989)', u'Jerry Maguire (1996)',

u'Liar Liar (1997)', u'Men in Black (1997)',

u'Mission: Impossible (1996)',

u'Monty Python and the Holy Grail (1974)',

u'Princess Bride, The (1987)', u'Pulp Fiction (1994)',

u'Raiders of the Lost Ark (1981)', u'Return of the Jedi (1983)',

u'Rock, The (1996)', u'Saint, The (1997)', u'Scream (1996)',

u'Silence of the Lambs, The (1991)', u'Star Trek: First Contact (1996)',

u'Star Wars (1977)', u'Terminator, The (1984)', u'Titanic (1997)',

u'Toy Story (1995)', u'Twelve Monkeys (1995)',

u'WillyWonka and the Chocolate Factory (1971)'],

dtype='object', name=u'movie title')

- ***<u>15 highly rated movies based on female rating</u>***:

gender	F	M
movie title		
Silence of the Lambs, The (1991)	4.320000	4.279310
Titanic (1997)	4.278846	4.231707
Star Wars (1977)	4.245033	4.398148
Fugitive, The (1993)	4.166667	4.011364

Godfather, The (1972) 4.133333 4.334416

Raiders of the Lost Ark (1981) 4.084211 4.301538

Princess Bride, The (1987) 4.044944 4.221277

Full Monty, The (1997) 4.020408 3.884793

Fargo (1996) 4.016000 4.201044

Return of the Jedi (1983) 4.008065 4.007833

Empire Strikes Back, The (1980) 3.978022 4.278986

Indiana Jones and the Last Crusade (1989) 3.923077 3.932806

Pulp Fiction (1994) 3.916667 4.100000

Rock, The (1996) 3.879121 3.634146

Monty Python and the Holy Grail (1974) 3.863636 4.120000

- ***Difference of mean ratings of movies based on male and female ratings***:

gender F ... diff

movie title ...

Air Force One (1997) 3.690476 ... -0.083919

Back to the Future (1985) 3.766667 ... 0.091026

Chasing Amy (1997) 3.819149 ... 0.026465

Contact (1997) 3.686131 ... 0.160643

E.T. the Extra-Terrestrial (1982) 3.839080 ... -0.008095

Empire Strikes Back, The (1980) 3.978022 ... 0.300964

English Patient, The (1996) 3.809211 ... -0.222584

Fargo (1996) 4.016000 ... 0.185044

Forrest Gump (1994) 3.772152 ... 0.108013

Fugitive, The (1993) 4.166667 ... -0.155303

Full Monty, The (1997) 4.020408 ... -0.135616

Godfather, The (1972) 4.133333 ... 0.201082

Independence Day (ID4) (1996) 3.688679 ... -0.332642

Indiana Jones and the Last Crusade (1989) 3.923077 ... 0.009729

Jerry Maguire (1996) 3.724138 ... -0.018914

[15 rows x 3 columns]

- ***Male and female are most deagree on movies***:

gender F ... diff

movie title ...

Independence Day (ID4) (1996) 3.688679 ... -0.332642

Rock, The (1996) 3.879121 ... -0.244975

Mission: Impossible (1996) 3.487179 ... -0.224022

English Patient, The (1996) 3.809211 ... -0.222584

Star Trek: First Contact (1996) 3.835616 ... -0.219178

Saint, The (1997) 3.247059 ... -0.169137

Willy Wonka and the Chocolate Factory (1971) 3.752688 ... -0.168997

Fugitive, The (1993) 4.166667 ... -0.155303

Full Monty, The (1997) 4.020408 ... -0.135616

Air Force One (1997) 3.690476 ... -0.083919

Titanic (1997) 4.278846 ... -0.047139

Silence of the Lambs, The (1991) 4.320000 ... -0.040690

Jerry Maguire (1996) 3.724138 ... -0.018914

E.T. the Extra-Terrestrial (1982) 3.839080 ... -0.008095

Return of the Jedi (1983) 4.008065 ... -0.000232

[15 rows x 3 columns]

5.3.2MEMORY BASED COLABORATIVE FILTERING :

WORK FLOW

i. Read u.data file
ii. Convert it into dataframe named df
iii. Split the dataset into training and test dataset where percentage of test example (test_size) is 0.25
iv. Create two user-item matrices one for training another for testing
v. Calculate the cosine similarity of the train data (user_similarity, item_similarity)
vi. Make prediction for item-based CF using formulae(user_prediction, item_prediction)
vii. Evaluate accuracy of user-based CF and item-based CF using MSE function

OUTPUT

User-based CF RMSE: 3.1278912151704135
Item-based CF RMSE: 3.4562654942596227

***5.3.3MODEL BASED COLABORATIVE FILTERING*:**

WORK FLOW of finding *root meansquare error*

i. Read in the file which contains the full dataset
ii. Split the dataset into training(train_data_matrix) and test(test_data_matrix) dataset where percentage of test example (test_size) is 0.25
iii. Find sparsity level of movie lens dataset
iv. Choose k=20
v. Find SVD components (**s,u,vt**) from test matrix
vi. Make a prediction (X_predict) by taking dot product of **s,u,vt**
vii. Calculate root mean square value between X_predict and test_data_matrix

OUTPUT

The sparsity level of MovieLens100K is 93.7%
User-based CF MSE: 2.710957747278451

WORK FLOW of Matrix Factorization

i. Build movie dicitionary(movies_dict) with line no as numpy movie id,its actual movie id as the key
ii. Read data from ratings file where each line of i/p file represents one tag applied to one movie by one user, which has the following format: userId,movieId,tag,timestamp
iii. Return a numpy array named numpy_arr
iv. Create P an initial matrix of dimension N x K, where is n is no of users and k is hidden latent features
v. Create Q an initial matrix of dimension M x K, where M is no of movies and K is hidden latent features
vi. Initialize steps variable which is the maximum number of steps to perform the optimisation, hardcoding the values
vii. Initialize alpha variable the learning rate, hardcoding the values
viii. Initialize beta variable the regularization parameter, hardcoding the values
ix. For each user, each item calculate the error of the element, second norm of P and Q for regularization, sum of norms
x. Compute the gradient from the error of each user, each item
xi. Compute total error
xii. Predictnumpy array of users and movie ratings

WORK FLOW of Recommendation

i. Read the rating file for the missing
ii. Get the mapping between movie names, actual movie id and numpy movie id
iii. Build predicted numpy movie id from the saved predicted matrix of user and movie ratings
iv. Create a dictionary of unrated movies for each user
v. Recommend top 25 unrated movies based on their the predicted score

OUTPUT

Top 25 movies recommendation for the user 1

Letters from Iwo Jima (2006) with Movie rating value 9.360346877845387
Eddie Murphy Raw (1987) with Movie rating value 9.236018165660075
Dear Wendy (2005) with Movie rating value 9.203698523462286
Talking About Sex (1994) with Movie rating value 8.956160676896394
Love! Valour! Compassion! (1997) with Movie rating value 8.947410581308626
Brave (2012) with Movie rating value 8.934428106940143
Wild at Heart (1990) with Movie rating value 8.929597840739088

Taking Chance (2009) with Movie rating value 8.865345347150067
Wonderland (1999) with Movie rating value 8.858284085776976
Goldfinger (1964) with Movie rating value 8.85507043606452
Advantageous (2015) with Movie rating value 8.841811638780884
Hurt Locker The (2008) with Movie rating value 8.827438714746878
Character (Karakter) (1997) with Movie rating value 8.814686428454374
Man from Elysian Fields The (2001) with Movie rating value 8.804616845028592
Elite Squad (Tropa de Elite) (2007) with Movie rating value 8.800233480476749
Andalusian Dog An (Chien andalou Un) (1929) with Movie rating value 8.782505251474689
All About the Benjamins (2002) with Movie rating value 8.773890359394672
Tortilla Soup (2001) with Movie rating value 8.758542219008639
Irma la Douce (1963) with Movie rating value 8.745150922721505
Lion in Winter The (1968) with Movie rating value 8.73719699262219
Mystery Science Theater 3000: The Movie (1996) with Movie rating value 8.730271624625018
Buried (2010) with Movie rating value 8.710702189343554
Ride the High Country (1962) with Movie rating value 8.709931728714334
Ice Harvest The (2005) with Movie rating value 8.700350486013615
Musketeer The (2001) with Movie rating value 8.666684616297236

Top 25 movies recommendation for the user 2

Star Maker The (Uomo delle stelle L') (1995) with Movie rating value 10.13880104020052
Haunting The (1999) with Movie rating value 10.09639976129478
Dersu Uzala (1975) with Movie rating value 9.986568480207652
Gran Torino (2008) with Movie rating value 9.633308061739772
Dot the I (2003) with Movie rating value 9.593458722493194
Love! Valour! Compassion! (1997) with Movie rating value 9.578484363839507
Repulsion (1965) with Movie rating value 9.525752185140883
No Country for Old Men (2007) with Movie rating value 9.489700718703212
Steam: The Turkish Bath (Hamam) (1997) with Movie rating value 9.464359147721858
Insurgent (2015) with Movie rating value 9.455806020643394
Great Santini The (1979) with Movie rating value 9.431101171056332
Bloodsport 2 (a.k.a. Bloodsport II: The Next Kumite) (1996) with Movie rating value 9.427678213008964
Dear Wendy (2005) with Movie rating value 9.322087684615997
Factotum (2005) with Movie rating value 9.312970454745962
To Catch a Thief (1955) with Movie rating value 9.25597696293851
Collapse (2009) with Movie rating value 9.20992322565514
Darkness (2002) with Movie rating value 9.177683020889742
Sword in the Stone The (1963) with Movie rating value 9.16489314214196
Upstream Color (2013) with Movie rating value 9.136938290335344
Taking Chance (2009) with Movie rating value 9.113214062168026
Green Hornet The (2011) with Movie rating value 9.107034466975747
Herbie Rides Again (1974) with Movie rating value 9.100971983950002
Dog Day Afternoon (1975) with Movie rating value 9.094729114359401
Midnight Cowboy (1969) with Movie rating value 9.088105650764051
Shower (Xizao) (1999) with Movie rating value 9.081735532919996

Top 25 movies recommendation for the user 3

Talking About Sex (1994) with Movie rating value 10.204998351509273
Suriyothai (a.k.a. Legend of Suriyothai The) (2001) with Movie rating value 9.947882319852024
Everyone Says I Love You (1996) with Movie rating value 9.865275533587132
Can't Buy Me Love (1987) with Movie rating value 9.856880582067522
Safe Conduct (Laissez-Passer) (2002) with Movie rating value 9.784257297830154
Death in Brunswick (1991) with Movie rating value 9.784003891609235
Poison Ivy II (1996) with Movie rating value 9.72720674424106
Low Life (1994) with Movie rating value 9.721290686061655
Jules and Jim (Jules et Jim) (1961) with Movie rating value 9.67255094340262
Making Plans for Lena (Non ma fille tu n'iras pas danser) (2009) with Movie rating value 9.617833833089582
Forbidden Planet (1956) with Movie rating value 9.59833795485311
Lion in Winter The (1968) with Movie rating value 9.490113114953642
Vertigo (1958) with Movie rating value 9.489959791757553
Eddie Murphy Raw (1987) with Movie rating value 9.478330514789848
Wild at Heart (1990) with Movie rating value 9.462678235822004
Trip to the Moon A (Voyage dans la lune Le) (1902) with Movie rating value 9.413287809691331
Sentinel The (2006) with Movie rating value 9.39122760646115
Great Santini The (1979) with Movie rating value 9.357351563700892
Evita (1996) with Movie rating value 9.332073442188022
Letters from Iwo Jima (2006) with Movie rating value 9.326944306809107
Alice in Wonderland (1951) with Movie rating value 9.321912754319111
No Country for Old Men (2007) with Movie rating value 9.318230556471313
Joe Kidd (1972) with Movie rating value 9.311247251165236
Wonderland (1999) with Movie rating value 9.288325162517323
Buried (2010) with Movie rating value 9.280888142401098

Memory-based techniques use the data (likes, ratings, etc) that we have to establish correlations (similarities?) between either users (Collaborative Filtering) or items (Content-Based Recommendation) to recommend an item i to a user u who's never seen it before. In the case of collaborative filtering, we get the recommendations from items seen by the user's who are closest to u, hence the term collaborative. In contrast, content-based recommendation tries to compare items using their characteristics (movie genre, actors, book's publisher or author… etc) to recommend similar new items.

In a nutshell, memory-based techniques rely heavily on simple similarity measures (Cosine similarity, Pearson correlation, Jaccard coefficient… etc) to match similar people or items together. If we have a huge matrix with users on one dimension and items on the other, with the

cells containing votes or likes, then memory-based techniques use similarity measures on two vectors (rows or columns) of such a matrix to generate a number representing similarity.
Model-based techniques on the other hand try to further fill out this matrix. They tackle the task of "guessing" how much a user will like an item that they did not encounter before. For that they utilize several machine learning algorithms to train on the vector of items for a specific user, then they can build a model that can predict the user's rating for a new item that has just been added to the system..

Popular model-based techniques are Bayesian Networks, Singular Value Decomposition, and Probabilistic Latent Semantic Analysis (or Probabilistic Latent Semantic Indexing). For some reason, all model-based techniques do not enjoy particularly happy-sounding names.

CHAPTER 6

CONCLUDING REMARKS:

Here we traversed through the process of making a basic recommendation engine in python. We stated by understanding the fundamentals of recommendations. Then we went on to load the Movie lens data set for the purpose of experimentation.

Subsequently we made a first model as a simple popularity model in which the most popular model in which the most popular movies are recommended for users. They can also find mean rating of a particular movie or search for top rated movies of particular genre or search for the movies by director name. But this lacked personalization. So we made another model based collaborative filtering and content based collaborative filtering.

We found that the the root means square error is less in model based filtering than both user based collaborative filtering and item based collaborative filtering . The important aspect is that the Collaborative Filtering model uses data (user_id, movie_id, rating) to learn the latent features. If there is little amount of data available model-based CF model will predict poorly, since it will be more difficult to learn the latent features.

Models that use both ratings and content features are called Hybrid Recommender Systems where both Collaborative Filtering and Content-based Models are combined. Hybrid recommender systems usually show higher accuracy than Collaborative Filtering or Content-based Models on their own: they are capable to address the cold-start problem better since if there is no ratings for a user or an item,one could use the metadata from the user or item to make a prediction.

We would like to propose another type of recommendation algorithm where factors like ratings, type of movie watched, age, occupation can be used to group users into "clusters" with similar viewing habits. A customer can belong to multiple clusters. Based on the cluster, we can then identify the movie characteristics that would be most appealing to the user. And we can recommend the user the top rated movie within that cluster.

➢ *REFERENCE:*

1. GauravArora, Ashish Kumar, Gitanjali Sanjay Devre, Prof. AmitGhumare , (2014) , MOVIE RECOMMENDATION SYSTEM BASED ON USERS' SIMILARITY , 4(3) , 765-766
2. Richhi F, Rokach L, Shapira B, Recommender Systems Handbook, DOI 10.1007/978-0-387-85820-3_1, © Springer Science+Business Media, LLC 2011, http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf
3. Manoj Kumar, D.K.Yadav, Ankur Sing, Vijay Kr. Gupta, 920150, International Journal of Computer Applications: A Movie Recommender System: MOVREC, 123(3), 7-10
4. Bhumika Bhatt, Prof. Premal J Patel, Prof. HetalGaudani, (2014), A Review Paper on Machine Learning Based Recommendation System, 2(4), 3955-3956
5. http://files.grouplens.org/datasets/movielens/ml-100k.zip
6. https://www.sciencedirect.com/science/article/pii/S1110866515000341
7. http://beyondvalence.blogspot.in/2014/09/python-and-pandas-part-2-movie-ratings.html
8. https://www.datacamp.com/community/tutorials/recommender-systems-python
9. http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/
10. https://blog.dominodatalab.com/recommender-systems-collaborative-filtering/
11. http://dataconomy.com/2015/03/an-introduction-to-recommendation-engines/
12. https://www.upwork.com/hiring/data/how-collaborative-filtering-works/
13. https://www.upwork.com/hiring/data/what-is-content-based-filtering/
14. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/memorybased.html
15. https://acodeforthought.wordpress.com/2016/12/26/building-a-simple-recommender-system-with-movie-lens-data-set/
16. https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/
17. http://enhancedatascience.com/2017/04/22/building-recommender-scratch/
18. https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html