# A HYBRID MOVIES RECOMMENDATION SYSTEM

A project submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology in

Information Technology

by

Aanchal Singh (Registration no:1602081001)

Abinash Mohapatra (Registration no: 1602081004)

Anisha Pradhan (Registration no: 1602081010)

Under the Supervision of

Dr. Manas Ranjan Senapati

**HOD Information Technology**


Department of Information Technology

Veer Surendra Sai University of Technology

Burla-768018, Odisha, India

July-2020


Department of Information Technology
Veer Surendra Sai University of Technology
Burla-768018, Odisha, India.

This is to certify that the project entitled "A Hybrid Movie Recommendation System" is being submitted by Ms. Aanchal Singh (Registration no:1602081001), Mr. Abinash Mohapatra (Registration no: 1602081004) and Ms. Anisha Pradhan (Registration no: 1602081010) to the department of Information Technology, Veer Surendra Sai University of Technology, Burla, for the award of the degree of Bachelor of Technology. It is an original research work carried out by them under my supervision and guidance.

In my opinion, the project has fulfilled all the requirements as per the regulations of this university and has reached the standard needed for submission. The results embodied in this project have not been submitted to any other university or institute for the award of any other degree or diploma.

Head of the department
Information Technology
VSSUT, Burla

Dr. Manas Ranjan Senapati
Supervisor

# Acknowledgement

We would like to express my deepest appreciation to all those who helped us during our project work. A special gratitude to our project guide, Dr. Manas Ranjan Senapati, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our minor project especially with his ideas who have invested his full effort in guiding the team in achieving the goal. His expertise and guidance proved to be valuable and provided a direction during every stage of the project.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of MCA ma'am Mrs. Anima Ma', who showed us how to lead the project and even guided us during our work.

We are also grateful to my classmates and friends with whom, the discussion sessions yielded various ideas and proved to be immensely fruitful while encountered with some dilemma.

Thank you, all teachers, you have been a great source of inspiration.

Lastly, we are thankful to our parents, without their support and inspiration, this project would not have been possible.

AanchalSingh(Registration no:1602081001)

AbinashMohapatra(Registration no: 1602081004)

AnishaPradhan(Registration no: 1602081010)

## ABSTRACT

This movie recommendation system adds a whole new dimension to the movie watching experience by providing real-time personalized movie recommendations to users. A recommendation system uses an algorithm to predict unknown information by utilizing knowledge available about the used and related items/information. It takes an approach where a user's own tastes are mixed with that of the entire community to generate meaningful results.

Most existing movie services like IMDB (www.imdb.com) do not personalize their recommendations but simply provide an overall rating for a movie. This significantly decreases the value of each recommendation as it does not cater to the individual movie preferences of the user. Unlike these systems, our Recommendation Engine will continually analyze individual user's movie preferences and recommend custom movie recommendations. The overall goal is to ease the movie discovery process. It is purely a movie recommendation service in that it offers a list of movie suggestions based on previous user ratings. It is designed not to search for movies but to discover them through our recommendation process.

The aim of this project focuses on finding the best type of technique used in recommendation system out of the 3 main types of technique. The 3 different type of techniques hence produce different results and accuracy.



Keywords: recommendation, content-based, collaborative filtering, hybrid model

# TABLE OF CONTENTS

# CHAPTER 1

## 1.1 OBJECTIVE

The main aim of this project is to develop a hybrid movie recommender system that:

- incorporates and enhances the properties of existing recommendation systems
- Adapt new approaches in order to decrease system runtime
- Reveal latent user and movie relations with great accuracy.

To achieve this, we have to evaluate all the existing recommendation techniques to find better models which can be fused with our hybrid recommendation model

## 1.2 INTRODUCTION

With the vast amount of data that the world has nowadays, institutions are looking for more and more accurate ways of using this data. Companies like Amazon use their huge amounts of data to give recommendations for users. Based on similarities among items, systems can give predictions for a new item's rating. Recommender systems use the user, item, and rating information to predict how other users will like a particular item. Recommender systems are now pervasive and seek to make a profit out of customers or successfully meet their needs. However, to reach this goal, systems need to parse a lot of data and collect information, sometimes from different resources, and predict how the user will like the product or item. The computation power needed is considerable. Also, companies try to avoid flooding customer mailboxes with hundreds of products each morning, thus they are looking for one email or text that will make the customer look and act. A recommendation system provides suggestions to the users through a filtering process that is based on user preferences and browsing history. The information about the user is taken as an input. This information reflects the prior usage of the product as well as the assigned ratings. A recommendation system is a platform that provides its users with various contents based on their preferences and likings. A recommendation system takes the information about the user as an input. The recommendation system is an implementation of the machine learning algorithm. Due to the advances in recommender systems, users constantly expect good recommendations. They have a low threshold for services that are not able to make appropriate suggestions. This has led to a high emphasis by tech companies on improving their recommendation systems. However, the problem is more complex than it seems.

The paper presents an overview of the field of recommender systems and describes the current generation of recommendation methods that are usually classified into the following three main categories: content-based, collaborative, and hybrid recommendation approaches. The paper also describes various limitations of current recommendation methods and discusses possible extensions that can improve recommendation capabilities and make recommender systems applicable to an even broader range of applications. These extensions include, among others, improvement of understanding of users and items, incorporation of the contextual information into the recommendation process, support for multi-criteria ratings, and provision of more flexible and less intrusive types of recommendations.

# CHAPTER 2

## 2.1. LITERATURE REVIEW

The theoretical background begins with the different types of Recommender Systemss, continues with the relations between RSs and customer experience and satisfaction, and ends with the adoption of RSs in the cultural sector.

**Types of RSs**: Recommender systems can be classified into three types: Collaborative filtering (CF), Content-based filtering (CB), and Hybrid system (Felício et al., 2016; Garcia, Sebastia, & Onaindia, 2011). A CF recommender system requires data on users' evaluation of purchase history to make suggestions (Lin, 2014). In contrast, CB recommender systems connect profiles of users' preferences with descriptions of relevant items (Mathew, Kuriakose, & Hegde, 2016). The hybrid systems integrate CF techniques with CB techniques or even with other techniques to optimize recommendations (Li, Xu, Wan, & Sun, 2018).

**RSs and Customer Experience**. Recommender systems play an important role in optimizing customer experience from finding to engaging with the products (Albadvi & Shahbazi, 2009; BarragánsMartínez et al., 2010). The most significant function of RSs is to predict pertinent services or products based on users' preferences (Lu, Wu, Mao, Wang, & Zhang, 2015; H.-R. Zhang & Min, 2016). To put it concisely, RSs have two key functions: to predict and to recommend the most relevant service or products (Ekstrand et al., 2011; Schafer et al., 2007).

**RSs and Customer Satisfaction**. Recommender systems have been proven as a means to increase customer satisfaction, maintain a long-term relationship, and improve financial performance (Briguez et al., 2014; Siu, Zhang, Dong, & Kwan, 2013). Particularly, the applications of RSs produce prosperous results in the field of cultural heritage, tourism, and leisure activities (Bartolini et al., 2016; Kabassi, 2013; A. Moreno et al., 2013). Movie and music RSs built by Netflix and Yahoo have noticeably improved the financial performance of these enterprises (Christensen & Schiaffino, 2011; Jannach, Resnick, Tuzhilin, & Zanker, 2016).

**RS Adoption in the Cultural Sector**. Regardless of a wide range of adoption of RSs in the cultural sector, studies on such research stream scatter all over the literature. As a consequence, it is challenging to take an overview of the big picture of the RS adoption among different cultural domains. Even though CF systems are proven to be effective for the entertainment and cultural domain, they raise issues related to user engagement and accuracy as recommendations inferred from a community may not be precise for an individual member (Kabassi, 2013; Villegas, Sánchez, Díaz-Cely, & Tamura, 2018). Another drawback of the CF approach is the insufficient amount of data input, which is defined as the cold start, impedes many SMEs/SMOs at the early stage of adoption (Kabassi, 2013; Park, Kim, Choi, & Kim, 2012). On the other hand, CB systems face the problem of identifying algorithms for effectively matching common

attributes between users and items (F. Deng, Ren, Qin, Huang, & Qin, 2018; Yao, Sheng, Segev, & Yu, 2013). The hybrid approach combing CF and CB has shown to outperform (van Capelleveen, Amrit, Yazan, & Zijm, 2019); yet, it requires a certain level of investment for information technology (IT) infrastructure which can be an obstacle for most SMEs/SMOs with financial constraints (Barragáns-Martínez et al., 2010; Park et al., 2012; Yao et al., 2013).

**Some more literature survey on recommendation systems**

- MOVREC [10] is a movie recommendation system presented by D.K. Yadav et al. based on collaborative filtering approach. The system also has a provision for user to select attributes on which he wants the movie to be recommended.

- Luis M Capos et al. [5] has analysed two traditional recommender systems i.e. content based filtering and collaborative filtering. As both of them have their own drawbacks he proposed a new system which is a combination of Bayesian network and collaborative filtering.

- A hybrid system has been presented by Harpreet Kaur et al. [9]. The system uses a mix of content as well as collaborative filtering algorithm.

- The user specific information or item specific information is clubbed to form a cluster by Utkarsh Gupta et al. [12] using chameleon.

- Urszula Kużelewska et al. [6] proposed clustering as a way to deal with recommender systems.

- Costin-Gabriel Chiru et al. [3] proposed Movie Recommender, a system which uses the information known about the user to provide movie recommendations. This system attempts to solve the problem of unique recommendations which results from ignoring the data specific to the user.

- To predict the difficulty level of each case for each trainee Hongli LIn et al. proposed a method called content boosted collaborative filtering (CBCF).The algorithm is divided into two stages, First being the content-based filtering nd the second being collaborative filtering. The CBCF algorithm involves the advantages of both CBF and CF, while at the same time, overcoming both their disadvantages.

Our project aims at comparing and implementing of the different techniques used in a ML based recommendation system. Which will conclude in finding out the best technique that shall give accurate recommendation.

# CHAPTER 3

## 3.1. OVERVIEW OF RECOMMENDATION SYSTEM

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relating to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read. "Item" is the general term used to denote what the system recommends to users.

A RS normally focuses on a specific type of item (e.g., CDs, or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item. RSs are primarily directed towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a Web site, for example, may offer. A case in point is a book recommender system that assists users to select a book to read. In the popular Web site, Amazon.com, the site employs a RS to personalize the online store for each customer. Since recommendations are usually personalized, different users or user groups receive diverse suggestions. Besides, there are also non-personalized recommendations. These are much simpler to generate and are normally featured in magazines or newspapers. Typical examples include the top ten selections of books, CDs, etc. While they may be useful and effective in certain situations, these types of non-personalized recommendations are not typically addressed by RS research. In their simplest form, personalized recommendations are offered as ranked lists of items. In performing this ranking, RSs try to predict what the most suitable products or services are, based on the user's preferences and constraints. To complete such a computational task, RSs collect from users their preferences, which are either explicitly expressed, e.g., as ratings for products, or are inferred by interpreting user actions. For instance, a RS may consider the navigation to a particular product page as an implicit sign of preference for the items shown on that page. RSs development initiated from a rather simple observation: individuals often rely on recommendations provided by others in making routine, daily decisions. For example it is common to rely on what one's peers recommend when selecting a book to read; employers count on recommendation letters in their recruiting decisions; and when selecting a movie to watch, individuals tend to read and rely on the movie reviews that a film critic has written and which appear in the newspaper they read. In seeking to mimic this behaviour, the first RSs applied algorithms to leverage recommendations produced by a community of users to deliver recommendations to an active user, i.e., a user looking for suggestions. The recommendations were for items that similar users (those with similar tastes) had liked. This approach is termed collaborative-filtering and its rationale is that if the active user agreed in the past with some users, then the other recommendations coming from these similar users should be relevant as well and of interest to the active user. As e-commerce Web sites began to develop, a pressing need

emerged for providing recommendations derived from filtering the whole range of available alternatives. Users were finding it very difficult to arrive at the most appropriate choices from the immense variety of items (products and services) that these Web sites were offering. The explosive growth and variety of information available on the Web and the rapid introduction of new e-business services (buying products, product comparison, auction, etc.) frequently overwhelmed users, leading them to make poor decisions.

## 3.2. GENERAL ARCHITECTURE OF RECOMMENDATION SYSTEM

Practically, recommender systems encompass a class of techniques and algorithms that can suggest "relevant" items to users. Ideally, the suggested items are as relevant to the user as possible, so that the user can engage with those items: YouTube videos, news articles, online products, and so on. Items are ranked according to their relevancy, and the most relevant ones are shown to the user. The relevancy is something that the recommender system must determine and is mainly based on historical data. If you've recently watched YouTube videos about elephants, then YouTube is going to start showing you a lot of elephant videos with similar titles and themes.
One common architecture for recommendation systems consists of the following components:

1. Candidate generation
2. Scoring
3. Re-ranking


Fig 3.2.1. Architecture of recommendation system

*Candidate Generation*: In this first stage, the system starts from a potentially huge corpus and generates a much smaller subset of candidates. For example, the candidate generator on YouTube reduces billions of videos down to hundreds or thousands. The model needs to evaluate queries quickly given the enormous size of the corpus. A given model may provide multiple candidate generators, each nominating a different subset of candidates. The following is the two common approaches for candidate generation:

1. Content-Based filtering
2. Collaborative filtering

*Scoring:* Next, another model scores and ranks the candidates to select the set of items (on the order of 10) to display to the user. Since this model evaluates a relatively small subset of items, the system can use a more precise model relying on additional queries.

*Re-ranking*: Finally, the system must take into account additional constraints for the final ranking. For example, the system removes items that the user explicitly disliked or boosts the score of fresher content. Re-ranking can also help ensure diversity, freshness, and fairness.

## 3.3. BASIC TEMINOLOGIES IN RECOMMENDATION SYSTEMS

RSs are information processing systems that actively gather various kinds of data to build their recommendations. Data is primarily about the items to suggest and the users who will receive these recommendations. But, since the data and knowledge sources available for recommender systems can be very diverse, ultimately, whether they can be exploited or not depends on the recommendation technique. Before going deep into the recommender system and its technique, one has to know some basic terminology related to it.

- **Item (also known as a document):** Items are the SS. It may be characterized by their complexity and their value or utility. The value of an item may be positive if the item is useful for the user, or negative if the item is not appropriate and the user made a wrong decision when selecting it. We note that when a user is acquiring an item, she will always incur a cost, which includes the cognitive cost of searching for the item and the real monetary cost eventually paid for the item.
- **Users:** Users of a RS, as mentioned above, may have very diverse goals and characteristics. To personalize the recommendations and the human-computer interaction, RSs exploit a range of information about the users. This information can be structured in various ways and again the selection of what information to model depends on the recommendation technique. Users can also be described by their behaviour pattern data, for example, site browsing patterns (in a Web-based recommender system), or travel search patterns (in a travel recommender system). Moreover, user data may include relations between users such as the trust level of these relations between users. A RS might utilize this information to recommend items to users that were preferred by similar or trusted users.
- **Query (also known as context):** Query is the information a system uses to make recommendations. Queries can be a combination of the following;
    1. user information
        1.1. the id of the user
        1.2. items that users previously interacted with
    2. additional context
        2.1. time of day
        2.2. the user's device

- **Transactions:** A transaction is referred as a recorded interaction between a user and the RS. Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using. For instance, a transaction log may contain a reference to the item selected by the user and a description of the context (e.g., the user goal/query) for that particular recommendation. If available, that transaction may also include explicit feedback the user has provided, such as the rating for the selected item. Sometimes transactions are also known as *'Embedding'*. Embedding is a mapping from a discrete set (in this case, the set of queries, or the set of items to recommend) to a vector space called embedding space. Many recommendation systems rely on learning an appropriate embedding representation of the queries and items.
- **Utility Matrix:** The data used in a recommendation system is divided in two categories: the users and the items. Each user likes certain items, and the rating value $r_{ij}$ (from 1 to 5) is the data associated with each user $i$ and item $j$ and represents how much the user appreciates the item. These rating values are collected in matrix, called utility matrix $R$, in which each row $i$ represents the list of rated items for user $i$ while each column $j$ lists all the users who have rated item $j$.

## 3.4. CLASSIFICATION OF RECOMMENDATION TECHNIQUES

The recommender systems can be categorized on several bases. In the literature, the categorization of the recommender systems are usually found on the following bases;

- Approaches used
- Area of application for which recommendation is made
- Data mining techniques applied, etc

But based on recommendation techniques recommendation system is classified in three categories. They are shown in figure 3.4.1 below:

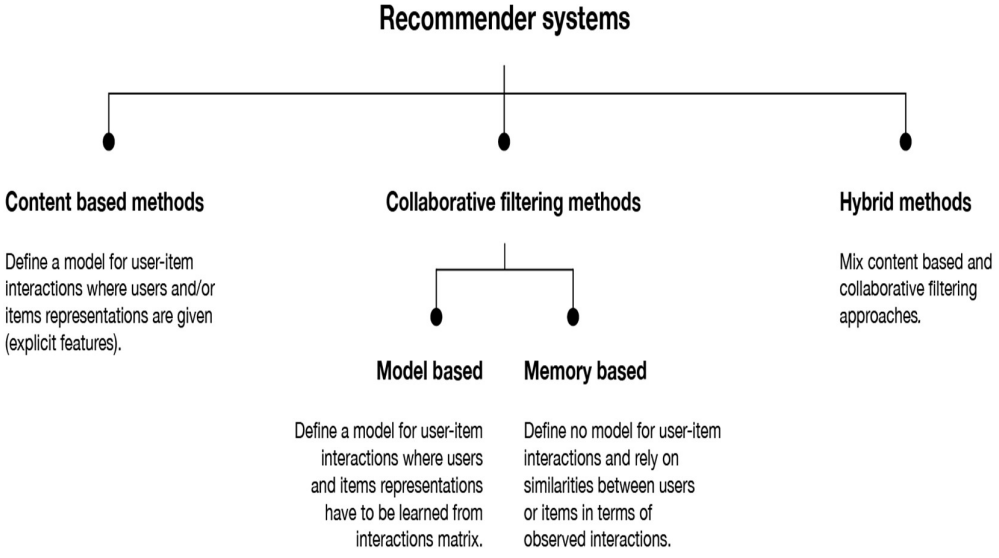


Fig 3.4.1 the 3 main categories of recommendation methods

## 3.4.1 CONTENT-BASED FILTERING

Content-based filtering, also referred to as cognitive filtering, recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. The user profile is represented with the same terms and built up by analyzing the content of items which have been seen by the user. In the context of movie recommendation system, we will consider genre as term of movie to see similarity. In this approach contents of the product are already **rated** based on the **user's preference (User Profile)**, while the genre of an item is an **implicit features** that it will be used to build **Item Profile**. An **item score** is then predicted by using both profiles and recommendation can be made.


Fig 3.4.1.1 Content Based Filtering

**Term Frequency-Inverse Document Frequency (TF-IDF)**

The concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) are used in information retrieval systems and also content based filtering mechanisms (such as a content based recommender). They are used to determine the relative importance of a document / article / news item /movie etc. Term Frequency (TF) and Inverse Document Frequency (IDF) TF is simply the frequency of a word in a document. IDF is the inverse of the document frequency among the whole corpus of documents. TF-IDF is used mainly because of two reasons: Suppose we search for The rise of analytics on Google, it is certain that 'The' will occur more frequently than analytics. However, the relative importance of analytics is higher than The from search query point of view. In such cases, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (document). We will consider genres as an important parameter to recommend user the movie he watches based on genres of movie user has already watched.

***TF-IDF Score***

$$TF - IDF\ Score = TF_{x,y} * IDF = TF_{x,y} * \log \frac{N}{df}$$

*, where $TF_{x,y}$ is the frequency of keyphrase X in the article Y,*
*N is the total number of documents in the corpus.*
*df is the number of documents containing keyphrase X*

**Term Frequency:** Frequency of the word in the current document to the total number of words in the document. It signifies the occurrence of the word in a document and gives higher weight when the frequency is more so it is divided by document length to normalize.

$$\mathrm{Tf}(t) = \frac{\text{Frequency occurence of term t in document}}{\text{Total number of terms in document}}$$

**Inverse Document Frequency:** Total Number of Documents to the frequency occurrence of documents containing the word. It signifies the rarity of the word as the word occurring the document is less the IDF increases. It helps in giving a higher score to rare terms in the documents**.**

$$\mathrm{Idf}(t) = log_{10}(\frac{\text{Total Number of documents}}{\text{Number of documents containing term t}})$$

**Cosine Similarity**

The dot product between two vectors is equal to the projection of one of them on the other. Therefore, the dot product between two identical vectors (i.e. with identical components) is equal to their squared module, while if the two are perpendicular (i.e. they do not share any directions), the dot product is zero. The dot product is important when defining the similarity, as it is directly connected to it. The definition of similarity between two vectors A and B is, in fact, the ratio between their dot product and the product of their magnitudes.

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

By applying the definition of similarity, this will be in fact equal to 1 if the two vectors are identical, and it will be 0 if the two are orthogonal. In other words, the similarity is a number bounded between 0 and 1 that tells us how much the two vectors are similar.

### 3.4.2 COLLABORATIVE FILTERING

The growth of the Internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering. Collaborative filtering is one of the techniques used for dealing with this problem.

The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with tastes similar to themselves. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis.

Collaborative filtering algorithms often require (1) users' active participation, (2) an easy way to represent users' interests, and (3) algorithms that are able to match people with similar interests.

Typically, the workflow of a collaborative filtering system is:

1. A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.
2. The system matches this user's ratings against other users' and finds the people with most "similar" tastes.
3. With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user (presumably the absence of rating is often considered as the unfamiliarity of an item)

A key problem of collaborative filtering is how to combine and weight the preferences of user neighbours. Sometimes, users can immediately rate the recommended items. As a result, the system gains an increasingly accurate representation of user preferences over time.

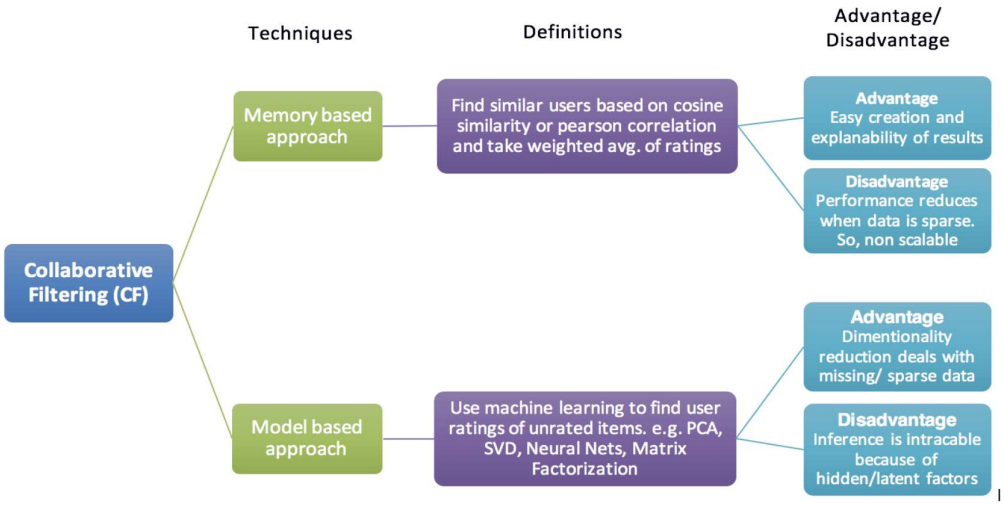



Fig 3.4.2.1 Classification of Collaborative filtering with definition, advantage/disadvantage

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

This falls under the category of **user-based** collaborative filtering. A specific application of this is the user-based Nearest Neighbour algorithm.

Alternatively, **item-based** collaborative filtering (users who bought x also bought y), proceeds in an item-centric manner:

1. Build an item-item matrix determining relationships between pairs of items
2. Infer the tastes of the current user by examining the matrix and matching that user's data.

Another form of collaborative filtering can be based on implicit observations of normal user behaviour (as opposed to the artificial behaviour imposed by a rating task). These systems observe what a user has done together with what all users have done (what music they have listened to, what items they have bought) and use that data to predict the user's behaviour in the future, or to predict how a user might like to behave given the chance. These predictions then have to be filtered through business logic to determine how they might affect the actions of a business system. For example, it is not useful to offer to sell somebody a particular album of music if they already have demonstrated that they own that music.

Relying on a scoring or rating system which is averaged across all users ignores specific demands of a user, and is particularly poor in tasks where there is large variation in interest (as in the recommendation of music). However, there are other methods to combat information explosion, such as web search and data clustering.

### 3.4.2.1. Memory-based Collaborative filtering

The memory-based approach uses user rating data to compute the similarity between users or items. Typical examples of this approach are neighbourhood-based CF and item-based/user-based top-N recommendations. For example, in user-based approaches, the value of ratings user ***u*** gives to item ***i*** is calculated as an aggregation of some similar users' rating of the item.

TYPES:

a) Item-Based Filtering
b) User-Based Filtering

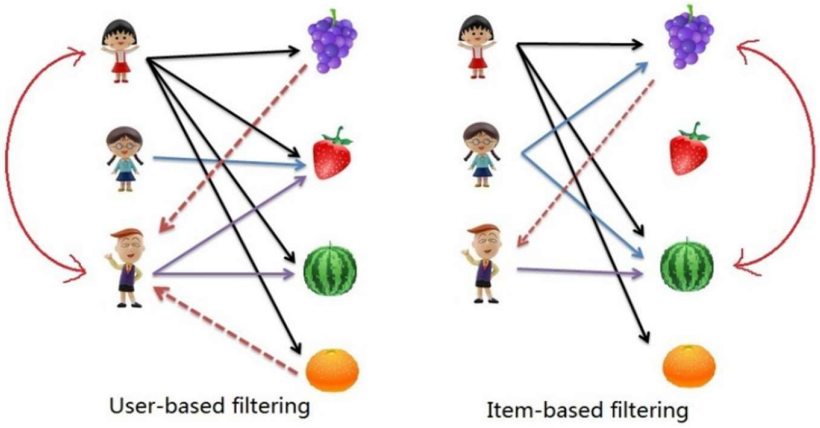



Fig 3.4.2.1.1 Types of memory-based collaborative filtering

In either scenario, we build a similarity matrix. For user-user collaborative filtering, the user-similarity matrix will consist of some distance metrics that measure the similarity between any two pairs of users. Likewise, the item-similarity matrix will measure the similarity between any two pairs of items.

**User-Based Filtering:**

The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. Here we find look alike users based on similarity and recommend movies which first user's look-alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every user pair information which takes time. Therefore, for big base platforms, this algorithm is hard to implement without a very strong parallelizable system. Well, **UB-CF** uses that logic and recommends items by finding similar users to the *active user* (to whom we are trying to recommend a movie). A

specific application of this is the user-based Nearest Neighbour algorithm. In other words, we are creating

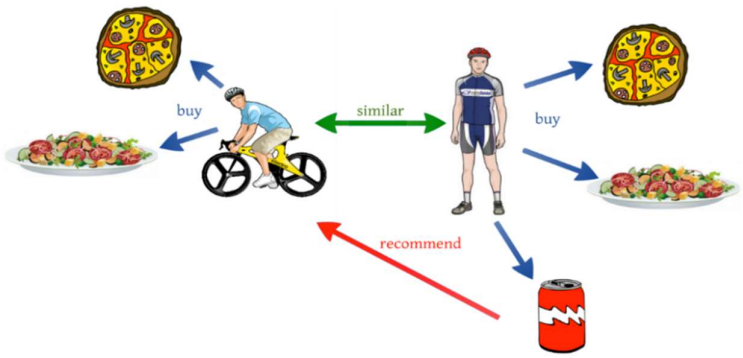



Fig 3.4.2.1.2 user-based collaborative filtering

a User-Item Matrix, predicting the ratings on items the active user has not seen, based on the other similar users. This technique is **memory-based**.

**Pros & Cons:**

| Advantages | Disadvantages |
|---|---|
| ▪ Easy to implement | • Sparsity: The percentage of people who rate items is very low |
| ▪ Context independent | • Scalability and New item: Just like the last point, new items will lack of ratings to create a solid ranking |
| ▪ Compared to other techniques, such as content based, it is more accurate | • Cold start: New users will have no to little information about them to be compared with other users |

Table 3.4.2.1.1 Pros and Cons of user-based CF

**Item-Based Filtering:**

Item based collaborative filtering, or item-based, or item-to-item, is a form of collaborative filtering for recommender systems based on the similarity between items calculated using people's ratings of those items.

Item-item collaborative filtering was invented and used by **Amazon.com**

**ITEM-BASED** collaborative filtering look for items that are like the articles that user has already rated and recommend most similar articles. But what does that mean when we say item-item similarity? In this case we don't mean whether two items are the same by attribute like Fountain pen and pilot pen are similar because both are pen. Instead, what similarity means is how people treat two items the same in terms of like and dislike.

It is quite like previous algorithm, but instead of finding user's look-alike, we try finding movie's look-alike. Once we have movie's look-alike matrix, we can easily recommend alike movies to user who have rated any movie from the dataset. This algorithm is far less resource consuming than user-user collaborative filtering. Hence, for a new user, the algorithm takes far lesser time than user-user collaborate as we don't need all similarity scores between users. And with fixed number of movies, movie-movie look alike matrix is fixed over time.

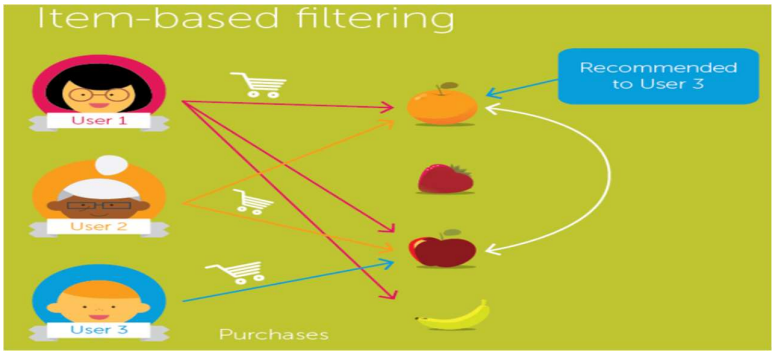



Fig 3.4.2.2. Item-based collaborative filtering

**Pros & Cons:**

| Advantages | Disadvantages |
|---|---|
| ▪ Item-item collaborative filtering had less error | ▪ It is very difficult to discover highly different items to recommend |
| ▪ It is less-dynamic model was computed less often and stored in a smaller matrix | ▪ Prediction tends to be less extreme (as its grounded with more data) |

Table 3.4.2.2. Pros and Cons of Item-based CF

### 3.4.2.2. Model-based Collaborative filtering

In this approach, collaborative filtering is realized by creating a machine learning model for predicting user's implicit ratings to the unrated items. Model-based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and visualize the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on other items. The model building process is performed by different *machine learning* algorithms such as **Bayesian network, clustering,** and **rule-based** approaches. There are various methods for implementation of model-based approach but generally it can break down into three categories as shown

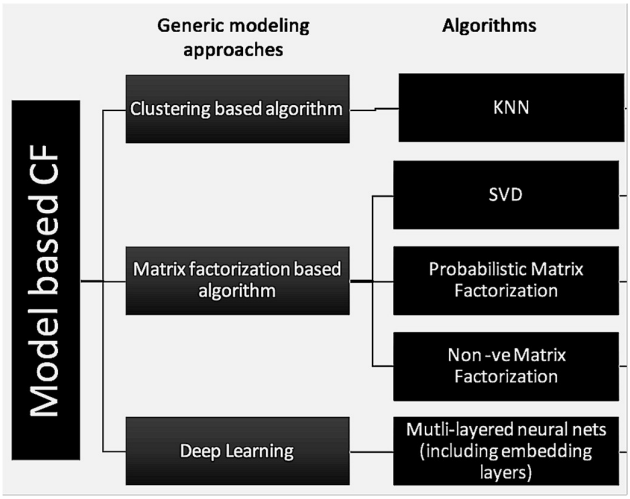



Fig 3.4.2.2.1 classification of model based collaborative filtering

Through this procedure, **dimensionality reduction** methods are mostly being used as supporting technique to boost robustness and accuracy of memory-based approach. In this sense, methods like singular value decomposition, principal component analysis, known as latent factor models, flatten user-item matrix into a low-dimensional representation in terms of latent factors. One advantage of using this approach is that instead of having a high dimensional matrix containing plentiful number of missing values we will be dealing with a much smaller matrix in lower-dimensional space. A diminished presentation could be utilized for either user-based or item-based neighborhood algorithms that are presented. There are several advantages with this paradigm. It handles the **sparsity** of the original matrix better than memory based ones. Also comparing similarity on the resulting matrix is much more scalable especially in dealing with large sparse datasets.

One of the model-based collaborative filtering is based on **matrix factorization (**MF) which has received greater exposure, mainly for dimensionality reduction and latent variable decomposition. Matrix factorization is widely used for recommendation systems where it can deal better with scalability and sparsity than Memory-based Collaborative Filtering. The purpose 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 (The sparsity level of MovieLens100K dataset is 98.3%), 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. This method of decomposing and restructuring of matrix is called as Singular Vector Decomposition (SVD).

**Singular Vector Decomposition**

The Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. SVD is a matrix factorization technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N). In the context of the recommender system, the SVD is used as a collaborative filtering technique. It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users. The factorization of this matrix is done by the singular value decomposition. It finds factors of matrices from the factorization of a high-level (user-item-rating) matrix. Singular Vector Decomposition is the method of decomposing a single matrix into three other matrices as follows:

$$A = U\Sigma V^T$$

Where A is input data matrix (user's ratings matrix)

U is left singular Orthogonal vector (user feature vector)

$\sum$ is the diagonal matrix of singular values (essentially weights/strengths of each concept)

$V^T$ is the right singular vectors (movie features matrix)

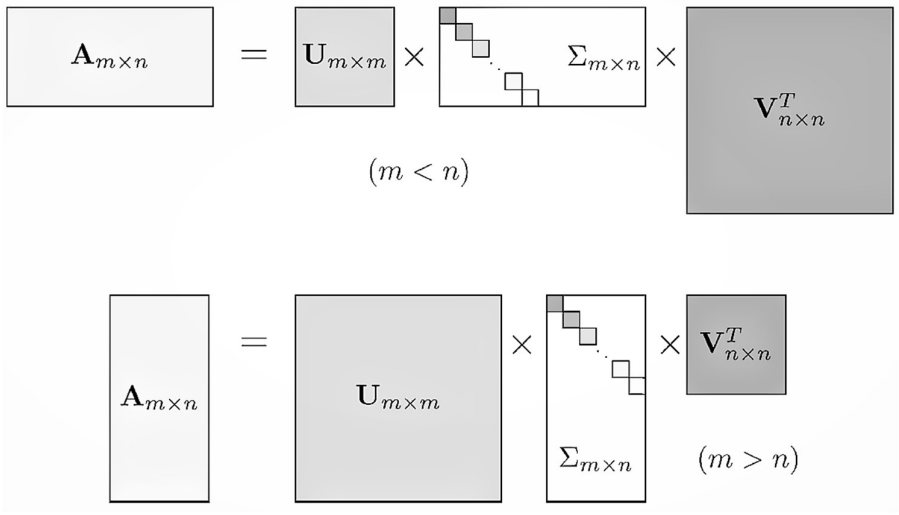



Fig 3.4.2.2.2. Matrix View Singular Vector Decomposition

The SVD decreases the dimension of the utility matrix $A$ by extracting its latent factors. It maps each user and each item into a $r$-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items.
Let each item be represented by a vector $x_i$ and each user is represented by a vector $y_u$. Then the expected rating given by the user to item can be represented as follows:

$$\hat{r}_{ui} = x_i^T . y_u$$

Here, $\hat{r}_{ui}$ is a form of factorization in singular value decomposition. The $x_i$ and $y_u$ can be obtained in a manner that the square error difference between their dot product and the expected rating in the user-item matrix is minimum. It can be expressed as:

$$Min(x,\ y) \sum_{(u,\ i) \in K} (r_{ui} - x_i^T . y_u)^2$$

In order to let the model generalize well and not overfit the training data, a regularization term is added as a penalty to the above formula.

$$Min(x,\ y) \sum_{(u,\ i) \in K} (r_{ui} - x_i^T . y_u)^2 + \lambda(||x_i||^2 + ||y_u||^2)$$

In order to reduce the error between the value predicted by the model and the actual value, the algorithm uses a bias term. Let for a user-item pair *(u, i)*, $\mu$ is the average rating of all items, $b_i$ is the average rating of item $i$ minus $\mu$ and $b_u$ is the average rating given by user $u$ minus $\mu$, the final equation after adding the regularization term and bias can be given as:

$$Min(x,\ y,\ b_i,\ b_u) \sum_{(u,\ i) \in K} (r_{ui} - x_i^T . y_u - \mu - b_i - b_u)^2 + \lambda(||x_i||^2 + ||y_u||^2 + b_i^2 + b_u^2)$$

The above equation is the main component of the algorithm which works for singular value decomposition-based recommendation system.

**SVD++**

The goal of a recommender system is to improve the predictive accuracy. In fact, the user will leave some implicit feedback information, such as historical browsing data, and historical rating data, on Web applications as long as any user has rated item, no matter what the specific rating value is. To a certain extent, the rating operation already reflects the degree of a user's preference for each latent factor. Therefore, the SVD++ model introduces the implicit feedback information based on SVD; that is, it adds a factor vector () for each item, and these item factors are used to describe the characteristics of the item, regardless of whether it has been evaluated. Then, the user's factor matrix is modelled, so that a better user bias can be obtained. Thus, the predictive rating of the SVD++ model is

$$\tilde{r}_{ui} = \mu + b_u + b_i + q_i^T \cdot \left( p_u + |R(u)|^{-1/2} \sum_{j \in R(u)} y_j \right),$$

where $R(u)$ is the number of items rated by user $u$.
To obtain the optimal user factor matrix $P$ and item factor matrix $q$ and, the regularized squared error can be minimized as follows. The objective function of the SVD++ model is

$$\min_{P,Q} \sum_{r_{ui} \in R} \left[ r_{ui} - \mu - b_u - b_i - q_i^T \cdot \left( p_u + |R(u)|^{-1/2} \sum_{j \in R(u)} y_j \right) \right]^2$$

### 3.4.2.3. Hybrid Recommendation Model

Several recommendation systems use a *hybrid* approach by combining collaborative and content-based methods, which helps to avoid certain limitations of content-based and collaborative systems. Different ways to combine collaborative and content-based methods into a hybrid recommender system can be classified as follows: (1) implementing collaborative and content-based methods separately and combining their predictions, (2) incorporating some content-based characteristics into a collaborative approach, (3) incorporating some collaborative characteristics into a content-based approach, and. All of the above approaches have been used by recommender systems researchers, as described below.

1. *Combining separate recommenders*. One way to build hybrid recommender systems is to implement separate collaborative and content-based systems. Then we can have two different scenarios. First, we can combine the outputs (ratings) obtained from individual recommender systems into one final recommendation using either a linear combination of ratings or a voting scheme. Alternatively, we can use one of the individual recommenders, at any given moment choosing to use the one that is "better" than others based on some recommendation "quality" metric.
2. *Adding content-based characteristics to collaborative models*. Several hybrid recommender systems, include the "collaboration via content" approach, are based on traditional collaborative techniques but also maintain the content-based profiles for each user. These content-based profiles, and not the commonly rated items, are then used to calculate the similarity between two users. This allows to overcome some sparsity-related problems of a purely collaborative approach, since typically not many pairs of users will have a significant number of commonly rated items.
3. *Adding collaborative characteristics to content-based models*. The most popular approach in this category is to use some *dimensionality reduction* technique on a *group* of content-based profiles. For example, let a recommendation system use latent semantic indexing (LSI) to create a collaborative view of a collection of user profiles, where user profiles are represented by term vectors, resulting in a performance improvement compared to the pure content-based approach.

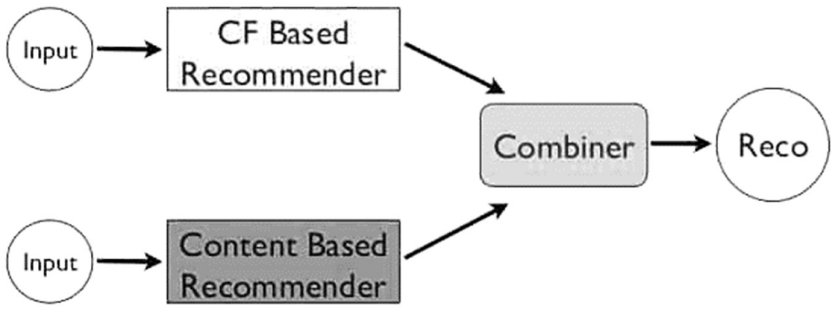



Fig 3.4.2.3.1. Hybrid Recommendation Model

Hybrid recommendation systems can also be augmented by knowledge-based techniques such as case-based reasoning, in order to improve recommendation accuracy and to address some of the limitations (e.g., new user, new item problems) of traditional recommender systems. For example, knowledge-based recommender system uses some domain knowledge about restaurants, cuisines, and foods (e.g., that "seafood" is not "vegetarian") to recommend restaurants to its users. The main drawback of knowledge-based systems is a need for knowledge acquisition – a well-known bottleneck for many artificial intelligence applications. However, knowledge-based recommendation systems have been developed for application domains where domain knowledge is readily available in some structured machine-readable form, e.g., as an ontology.

## 3.5 DISTANCE MEASURES

In Data Mining sense, the similarity measure is a distance with dimensions describing object features. That means if the distance among two data points is **small** then there is a **high** degree of similarity among the objects and vice versa. The similarity is **subjective** and depends heavily on the context and application. For example, similarity among vegetables can be determined from their taste, size, colour etc.
Most clustering approaches use distance measures to assess the similarities or differences between a pair of objects. In our recommendation techniques we have also used some of the similarity measures. The most popular distance measures used are:

### 3.5.1. Euclidean Distance

Euclidean distance is considered the traditional metric for problems with geometry. It can be simply explained as the **ordinary distance** between two points. It is one of the most used algorithms in the cluster analysis. One of the algorithms that use this formula would be **K-mean**. Mathematically it computes the **root of squared differences** between the coordinates between two objects. The general formula for Euclidean distance is as follows:

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

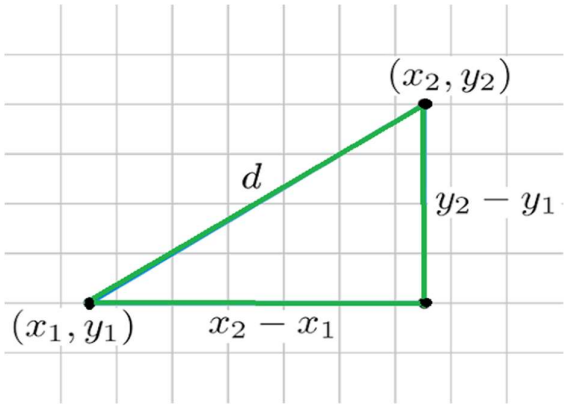



Fig 3.5.1.1. Euclidean distance between two points

### 3.5.2 Cosine Similarity Measure

Cosine similarity is a metric used to determine how similar two entities are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. *Mathematically, if 'a' and 'b' are two vectors, cosine equation gives the angle between the two is given as below:*

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

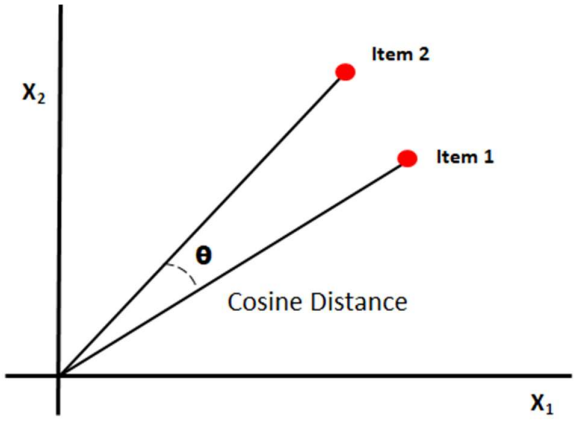


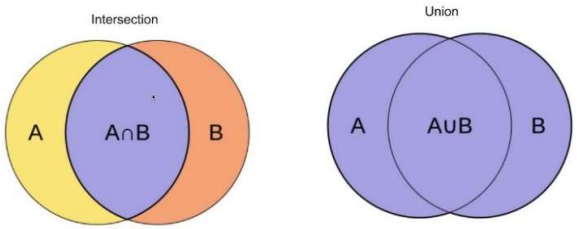

Fig 3.5.2 Cosine Similarity between item 1 and item 2

### 3.5.3 Jaccard Similarity index

The Jaccard similarity index (sometimes called the Jaccard similarity *coefficient*) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations. Although it's easy to interpret, it is extremely sensitive to small samples sizes and may give erroneous results, especially with very small samples or data sets with missing observations.

The Jaccard distance measures the similarity of the two data set items as the **intersection** of those items divided by the **union** of the data items.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$




$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Fig 3.5.3 Jaccard Co-efficient

### 3.5.4 Pearson Correlation

Correlation between sets of data is a measure of how well they are related. The most common measure of correlation in stats is the Pearson Correlation. The full name is the ***Pearson Product Moment Correlation (PPMC).*** It shows the linear relationship between two sets of data. In simple terms, it answers the question, *Can I draw a line graph to represent the data?* The formula for Pearson correlation similarity is give below:

$$similarity(A,B) = \frac{\text{cov}(A,B)}{\sigma_A \sigma_B} = \frac{\sum_{i=1}^{n}(A_i - \overline{A})(B_i - \overline{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \overline{A})^2 (B_i - \overline{B})^2}}$$

Here, $\bar{A}$ is mean of feature A and $\bar{B}$ is mean of feature B

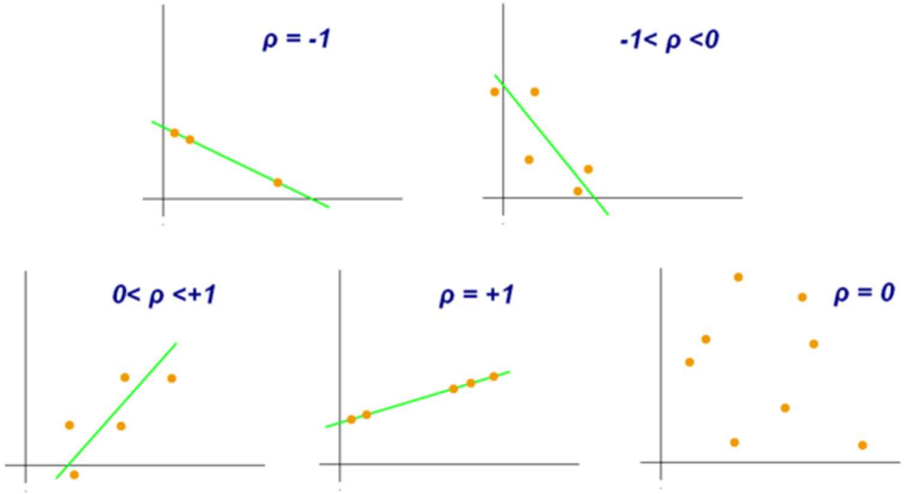



Fig 3.5.4 Examples of scatter diagrams with different values of Pearson correlation coefficient (ρ)

# CHAPTER 4: IMPLEMENTATION

## 4.1. PROBLEM STATEMENT

The Objective of this project is to develop a hybrid movie recommender system that:

- incorporates and enhances the properties of existing recommendation systems
- Adapt new approaches in order to decrease system runtime
- Reveal latent user and movie relations with great accuracy.

To achieve this, we have to evaluate all the existing recommendation techniques to find better models which can be fused with our hybrid recommendation model

## 4.2. EXPERIMENTAL ENVIRONMENT

Processor: 2.6 GHz Intel Core i5
Memory: 8 GB 1600 MHz DDR3
Operation System: Windows 10
Language: Python
Platform: Anaconda Jupyter Notebook and Google Colab

## 4.3. PYTHON LIBRARIES

Python is a popular and powerful interpreted language. Python is a complete language and platform that is used for both research and development and developing production systems. Here we have used Google Colab Environment, which provides free runtime GPU and it allows to install additional third-party Python libraries and uploading and using data files to your Colab environment.

**Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The library is highly optimized for performance.

The library contains different features, such as:

a) Data frame object for data manipulation with integrated indexing.
b) Data set joining and merging.
c) Reshaping and pivoting data set, and many more.

Pandas was developed by Wes McKinney and started working on pandas in 2008 while at AQR Capital Management out of the need for a high performance, flexible tool to perform quantitative analysis on financial data. Before leaving AQR he was able to convince management to allow him to open source the library.

**Numpy:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Seaborn:** Seaborn is a data visualization Python library based on matplotlib. Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. Here is some of the functionality that seaborn offers:

- A dataset-oriented API for examining relationships between multiple variables
- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds dependent variables
- Convenient views onto the overall structure of complex datasets
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- Concise control over matplotlib figure styling with several built-in themes
- Tools for choosing color palettes that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

**Matplotlib: Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

It was developed by John D Hunter, as an active development community, under a BSD style license. Pyplot is a matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use

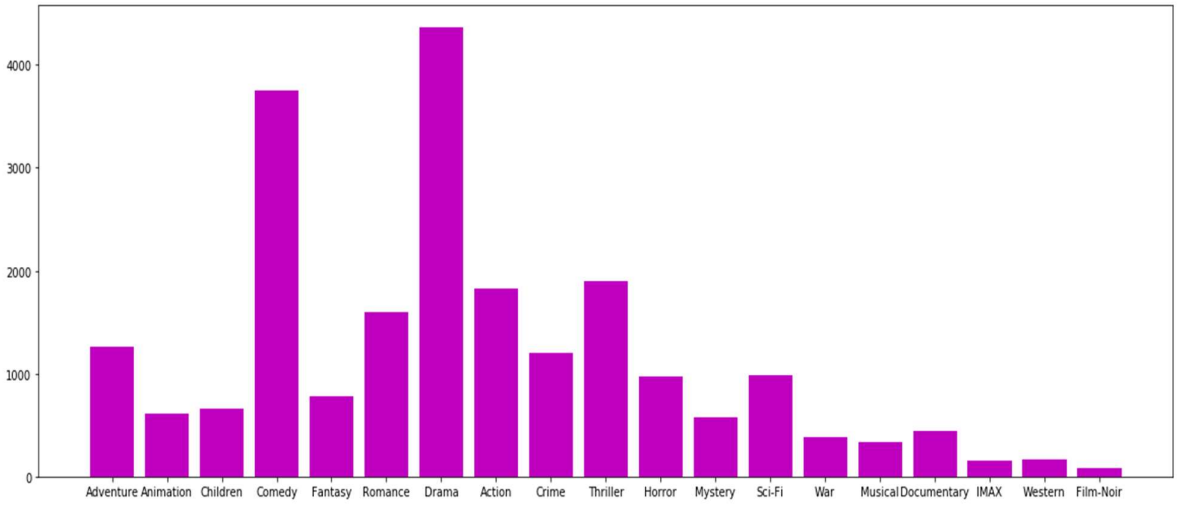
Python, and the advantage of being free and open-source. It plots different graph such as line plot, histogram, scatter plot, 3D plot, etc.

## 4.4. DATA COLLECTION AND LOADING

For this research project, we have used MovieLens 100K dataset. MovieLens datasets were collected by the GroupLens Research Project at the University of Minnesota. This dataset consists of: 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies. The data was collected through the MovieLens website (movielens.umn.edu).

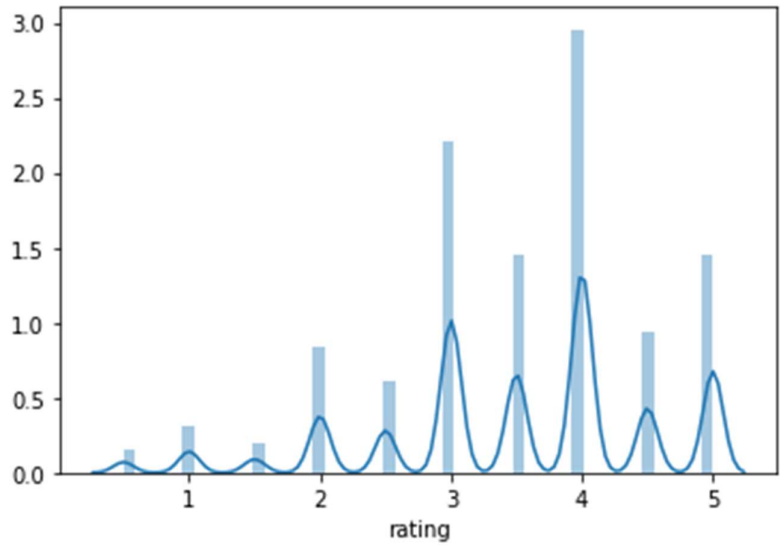
## 4.5. DATA ANALYSIS AND FORMATION OF UTILITY MATRIX

- The MovieLens 110K dataset is loaded from the movielens.umn.edu. The dataset consists of 100,000 ratings (1-5) from 943 users on 1682 movies.
- The movie data frame contains 19 genres (features) and in this context we will consider genre as the term of the movie to see similarity.
- The rating data frame contains ratings(1-5) of each movie by each user.
- Shape of both the frames are Movies data frame (9742,6) and Rating data frame (100836,4).
- In the movie data frame, the most popular genre is Drama.




Graph 4.5.1 Most Popular Genre

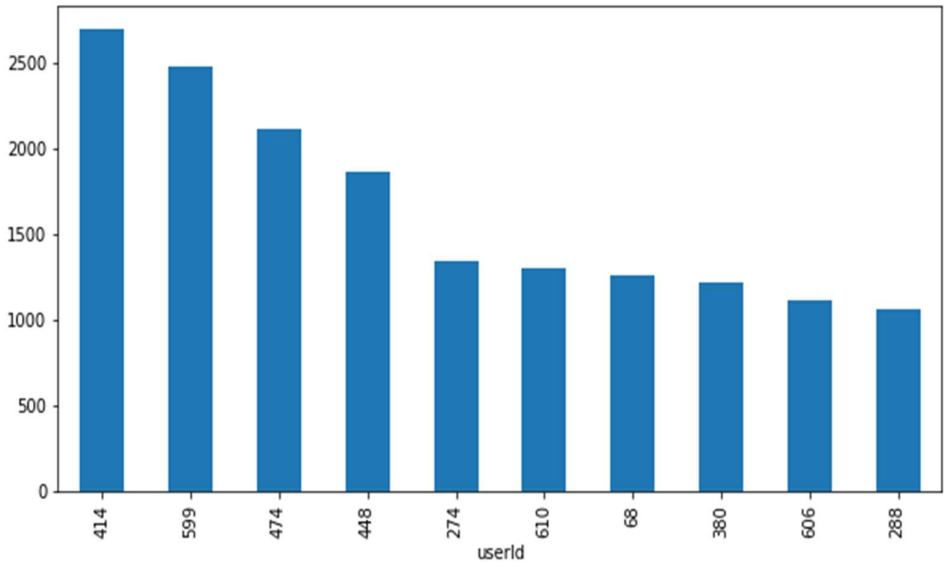
- In the rating data frame, most of the users rated Good(4) the movies.




Graph 4.5.2 Rating based on users

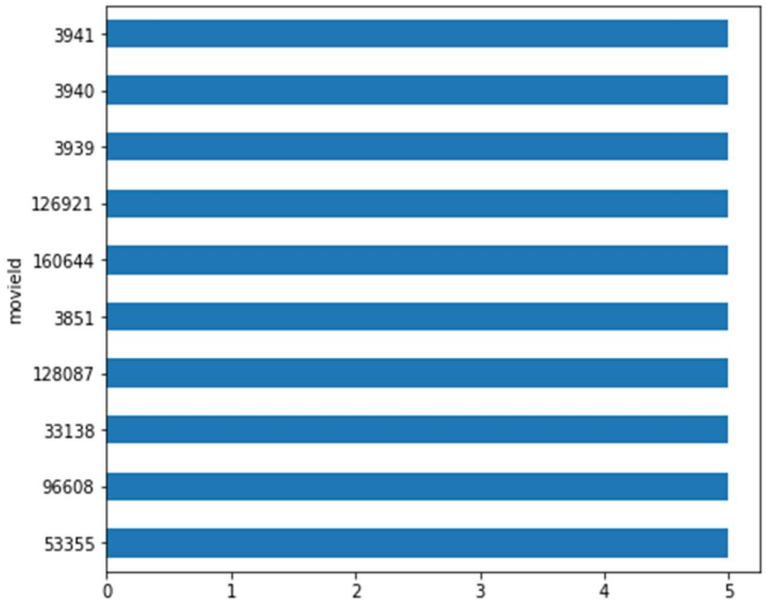
- Both the rating and movies data frame are inner merged on 'movieId' to form a single data frame so that grouping can be made a little complex.

- The ratings are grouped based on users to find out top 10 users who have rated most of the movies.

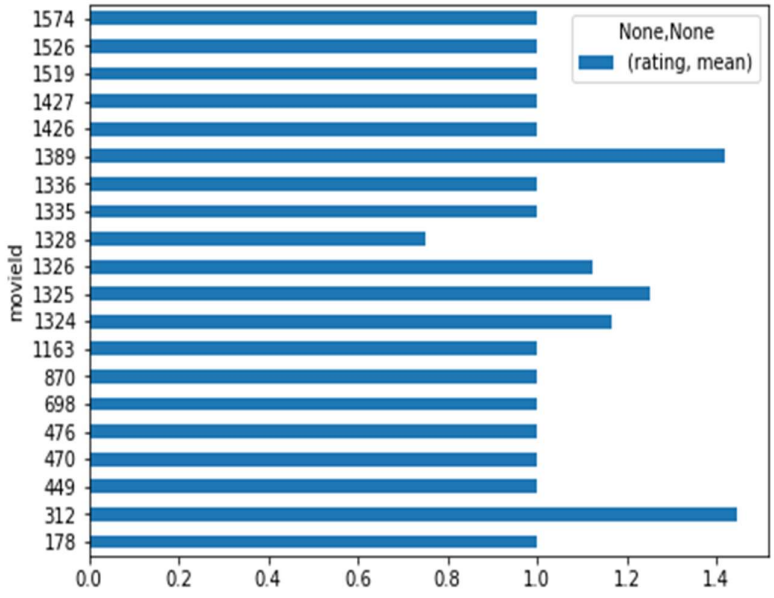



Graph 4.5.3 Top 10 users who have rated most of the movies

- Again, the ratings are grouped based on movies to find out the movie with high average rating and low average rating.




Graph 4.5.4 Movies with high average rating




Graph 4.5.5 Movies with low average rating

- After the exploratory data analysis the utility matrix is formed.

| | user | Toy Story (1995);1 | GoldenEye (1995);2 | Four Rooms (1995);3 | Get Shorty (1995);4 | Copycat (1995);5 | Twelve Monkeys (1995);7 | Babe (1995);8 | Dead Man Walking (1995);9 | Richard III (1995);10 | ... | Cool Runnings (1993);1035 | Hamlet (1996);1039 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 5 | 3 | 4 | 3 | 3 | 4 | 1 | 5 | 3 | ... | 0 | 0 |
| **1** | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | ... | 0 | 0 |

2 rows × 604 columns

Fig 4.5.1 Utility Matrix

## 4.6. IMPLEMENTATION OF CONTENET-BASED FILTERING

Content-based filtering, also referred to as cognitive filtering, recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. The user profile is represented with the same terms and built up by analyzing the content of items which have been seen by the user. In the context of movie recommendation system, we will consider genre as term of movie to see similarity. In this approach contents of the product are already **rated** based on the **user's preference (User Profile)**, while the genre of an item is an **implicit features** that it will be used to build **Item Profile**. An **item score** is then predicted by using both profiles and recommendation can be made.

### Term Frequency-Inverse Document Frequency (TF-IDF)

The concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) are used in information retrieval systems and also content based filtering mechanisms (such as a content based recommender). They are used to determine the relative importance of a document / article / news item /movie etc. Term Frequency (TF) and Inverse Document Frequency (IDF) TF is simply the frequency of a word in a document. IDF is the inverse of the document frequency among the whole corpus of documents. TF-IDF is used mainly because of two reasons: Suppose we search for The rise of analytics on Google, it is certain that 'The' will occur more frequently than analytics. However, the relative importance of analytics is higher than The from search query point of view. In such cases, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (document). We will consider genres as an important parameter to recommend user the movie he watches based on genres of movie user has already watched.

$$\mathrm{Tf}(t) = \frac{\text{Frequency occurence of term t in document}}{\text{Total number of terms in document}}$$

$$\mathrm{Idf}(t) = log_{10}\left(\frac{\text{Total Number of documents}}{\text{Number of documents containing term t}}\right)$$

**The steps we took into this technique are:**

1. Collect the movie and rating data frame from the MovieLens dataset and specify the term of movie to see similarity.
2. Inner merge both movie and ratings data frame on movieId.

3. Encode the data by calculating TF-IDF for each item. It is calculated as follows:

***TF-IDF Score***

$$TF - IDF\ Score = TF_{x,y} * IDF = TF_{x,y} * \log \frac{N}{df}$$

*, where $TF_{x,y}$ is the frequency of keyphrase X in the article Y,*
*N is the total number of documents in the corpus.*
*df is the number of documents containing keyphrase X*

A vector encoded document will form, in which each element in the vector represents a TF-IDF weight associated with a term in a document.

4. To measure the distance between two movies, here we used Cosine Similarity. It is calculated as:

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

5. For evaluating model we have clustered movie based on groups of genres with the KNN classifier. The classifier label returned by KNN classifier to the movies recommended by the content based filtering is compare to the classifier label of the movie on which model recommends, the hit and error are calculated accordingly to measure accuracy.

## 4.7. IMPLEMENTAION OF MEMORY-BASED COLLABORATIVE FILTERING

Content based filtering approach which is based on a comparison between the content of the items and a user profile had some issues that can be optimized in memory based collaborative filtering. Hence memory based collaborative system is preferred over content-based approach. Some of the reasons are given below:

- Not required to understand item content: The content of the items does not necessarily tell the whole story, such as movie type/genre, and so on.
- No item cold-start problem: Even when no information on an item is available, we still can predict the item rating without waiting for a user to purchase it.
- Captures the change in user interests over time: Focusing solely on content does not provide any flexibility on the user's perspective and their preferences.

Memory-based algorithms utilize the entire user-item database to generate a prediction. These systems employ statistical techniques to find a set of users, known as *neighbors* that have a history of agreeing with the target user. Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or *top*-N recommendation for the active user. Typical examples of this

approach are **neighborhood-based CF** and **item-based/user-based top-N recommendations**. There are two types of memory based collaborative filtering technique:

1. Item based collaborative filtering
2. User based collaborative filtering

In either scenario, we build a similarity matrix. For user based collaborative filtering, the user-similarity matrix will consist of some distance metrics that measure the similarity between any two pairs of users. Likewise, the item-similarity matrix will measure the similarity between any two pairs of items.

The nearest user or items are calculated only by using **Cosine similarity, Jaccard Similarity or Pearson correlation coefficients**, which are only based on arithmetic operations. Since we are finding the cosine of two vectors the output will always range from 0 to 1, since the ratings are all positive. We will now see how we can use the Cosine Similarity measure to determine how similar the movies are.

**Predicting Ratings**: In order to predict a rating for an item for an active user, we need to find all weights between the active user and all other users. We then take all non-zero weights and have each other user "vote" on what they think the active user should rate the item. Those with higher weights will matter more in the voting process. Once these votes are tallied, we have a predicted vote.

**User based collaborative filtering**

In user-based, similar users which have similar ratings for similar items are found and then target user's rating for the item which target user has never interacted is predicted

**The steps we took in this technique are:**

1. Collect the entire user-item database and specify the target user.
2. Create a sparse matrix where we write user-item ratings in a matrix form, with this one to one mapping between movies, user and their rating is done.

| Item / User | I1 | I2 | I3 | ... | In |
|---|---|---|---|---|---|
| U1 | $R_{1,1}$ | $R_{1,2}$ | $R_{1,3}$ | ... | $R_{1,n}$ |
| U2 | $R_{2,1}$ | $R_{2,2}$ | $R_{2,3}$ | ... | $R_{2,n}$ |
| U3 | $R_{3,1}$ | $R_{3,2}$ | $R_{3,3}$ | ... | $R_{3,n}$ |
| ... | ... | ... | ... | ... | ... |
| Um | $R_{m,1}$ | $R_{m,2}$ | $R_{m,3}$ | ... | $R_{m,n}$ |

**Table 4.7.1 User-Item rating matrix for user-item CF**

3. Find similar users who have similar ratings to target user (can be more than one) using distance similarity matrices (You can use the pairwise distances function from sklearn to calculate the cosine similarity).
4. Find a pair of users which are most similar.

Fig 4.7.1. Neighborhood Formation

5. Extract the items which target user never interacted.
6. Predict the ratings of unobserved items for target user.

Prediction formula for user based:

$$P_{u,i} = A_u + \frac{\sum_{w=1}^{n}(R_{w,i} - A_w) \times sim(u,w)}{\sum_{w=1}^{n} sim(u,w)}$$

Where u is the target user

i is the target item

$A_u$ is the average ratings of the target user $u$ to all other rated items

$A_w$ is the average ratings of the neighbour user $w$ to all other rated items

$R_w$,i is the rating of the neighbour user $w$ to the target item $i$.

$si(u,w)$ is the similarity of the target User $u$ and the neighbour user $w$.

$n$ is the total number of neighbours.

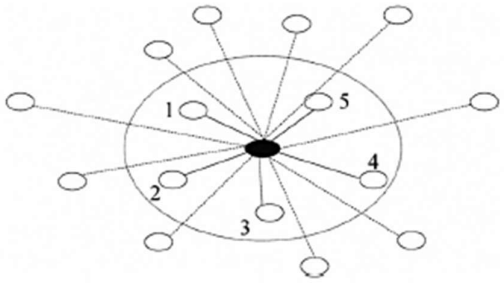
7. If the predicted ratings are above the threshold, then recommend them to target user

**Item based collaborative filtering**

Item-based models find similar items to items which target user already rated or interacted.

**The steps we took in this technique are:**

1. Collect the entire user-item database and specify the target user.



2. Create a sparse matrix where we write user-item ratings in a matrix form, with this one to one mapping between movies, user and their rating is done.

| Item / User | I1 | I2 | I3 | ... | In |
|---|---|---|---|---|---|
| U1 | $R_{1,1}$ | $R_{1,2}$ | $R_{1,3}$ | … | $R_{1,n}$ |
| U2 | $R_{2,1}$ | $R_{2,2}$ | $R_{2,3}$ | … | $R_{2,n}$ |
| U3 | $R_{3,1}$ | $R_{3,2}$ | $R_{3,3}$ | … | $R_{3,n}$ |
| ... | … | … | … | … | … |
| Um | $R_{m,1}$ | $R_{m,2}$ | $R_{m,3}$ | … | $R_{m,n}$ |

**Table 4.7.2. User-Item rating matrix for item-item CF**

3. Find similar items which have similar ratings with items target user rated using distance similarity matrices. (You can use the pairwise distances function from sklearn to calculate the cosine similarity).

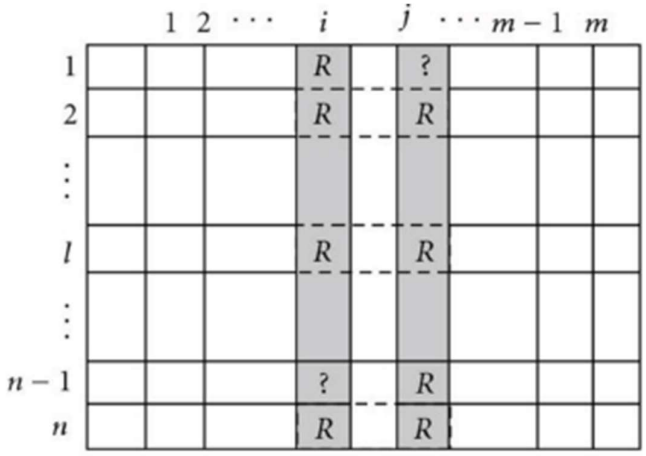



**Fig 4.7.2. Item based similarity computation**

4. Predict the ratings for similar items.

Prediction formula for item based:

$$P_{u,i} = \frac{\sum_{j=1}^{n} R_{u,j} \times sim(i,j)}{\sum_{j=1}^{n} sim(i,j)}$$

Where u is the target user

i is the target item

$R_{u,j}$ is the rating of the target user $u$ to the target Item $i$.

$si(i, j)$ is the weighted similarity of the target Item $i$ and the neighbor Item $j$.

*n* is the total number of neighbor items

5. If the predicted ratings are above the threshold, then recommend them to target user.

## 4.8. IMPLEMENTAION OF MODEL-BASED COLLABORATIVE FILTERING

Memory-based collaborative filtering approaches that compute distance relationships between items or users have these two major issues:

- It doesn't scale particularly well to massive datasets, especially for real-time recommendations based on user behaviour similarities—which takes a lot of computations.
- Ratings matrices may be overfitting to noisy representations of user tastes and preferences. When we use distance based "neighbourhood" approaches on raw data, we match to sparse low-level details that we assume represent the user's preference vector instead of the vector itself.

Thus, we need to apply Dimensionality Reduction technique to derive the tastes and preferences from the raw data, otherwise known as doing **low-rank matrix factorization**.

Model-based Collaborative Filtering is based on **matrix factorization** (MF) which has received greater exposure, mainly for dimensionality reduction and latent variable decomposition. Matrix factorization is widely used for recommendation systems where it can deal better with scalability and sparsity than Memory-based Collaborative Filtering. When you have a very sparse matrix (The sparsity level of MovieLens100K dataset is 98.3%), 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. We have chosen models inspired by Singular Value Decomposition (SVD) to predict the user-item ratings matrix. SVD models are very popular primarily to their scalable nature and accuracy. **SVD** is a method of decomposing a matrix into three other matrices:

$$A = U\Sigma V^T$$

A is the input data matrix (user's ratings), $U$ is the left singular vectors (user features matrix), $\sum$ is the diagonal matrix of singular values (essentially weights/strengths of each concept), and $V^T$ is the right singular vectors (movie features matrix). $U$ and $V^T$ are column orthogonal, and represent different things. $U$ represents how much users like each feature and $V^T$ represent how relevant each feature is to each movie.

To get the lower rank approximation, we take these matrices and keep only the top k features, which can be thought of as the underlying tastes and preferences vectors.

To build a robust recommender system, we need to develop models which factor in both explicit and implicit user feedback. For our Movielens dataset, a less obvious kind of implicit data does exist. The dataset doesn't only tell us the rating values, but also which movies users' rate, regardless of how they rated these movies. In other words, a

user implicitly tells us about her preferences by choosing to voice her opinion and vote a (high or low) rating. This reduces the rating matrix into a binary matrix, where 1 stands for rated, and 0 for not rated. Admittedly, this binary data is not as vast and independent as other sources of implicit feedback could be. Nonetheless, we have found that incorporating this kind of implicit data which inherently exist in every rating-based recommender system significantly improves prediction accuracy. **SVD++** factors in this implicit feedback. Mathematically SVD++ can be represented as shown.

$$\tilde{r}_{ui} = \mu + b_u + b_i + q_i^T \cdot \left( p_u + |R(u)|^{-1/2} \sum_{j \in R(u)} y_j \right),$$

where $R(u)$ is the number of items rated by user $u$.

Where the *yj* terms are a new set of item factors that capture implicit ratings. Here, an implicit rating describes the fact that a user u rated an item *j*, regardless of the rating value.

***Dimensionality reduction***

A typical machine learning problem might have several hundred or more variables, while many machine learning algorithms will break down if presented with more than a few dozen. This makes singular value decomposition indispensable in ML for variable reduction. We have already seen in how an SVD with a reduced number of singular values can closely approximate a matrix. This can be used for data compression by storing the truncated forms of U, $\Sigma$, and V in place of A and for variable reduction by replacing A with U. Below are some of the advantages of dimensionality reduction:

- We can discover hidden correlations / features in the raw data.
- We can remove redundant and noisy features that are not useful.
- We can interpret and visualize the data easier.
- We can also access easier data storage and processing.

| Item / User | I1 | I2 | I3 | I4 | I5 |
|---|---|---|---|---|---|
| U1 | 2.0 | 4.0 | 3.0 | 3.0 | 3.0 |
| U2 | | | 1.0 | 3.0 | |
| U3 | 5.0 | 1.0 | 5.0 | 5.0 | |
| U4 | 4.0 | | 3.0 | | |

Table 4.8.1 User-Item Matrix before dimensionality reduction

| Item / User | I1 | I3 | I4 |
|---|---|---|---|
| U1 | 2.0 | 3.0 | 3.0 |
| U2 | | 1.0 | 3.0 |
| U3 | 5.0 | 5.0 | 5.0 |
| U4 | 4.0 | 3.0 | |

Table 4.8.2 User-Item matrix after dimensionality reduction

For instance, as seen in [Table 4.8.1], each item can get up to a maximum of 4 ratings by users. Item I2 has 2 ratings and Item I5 has 1 rating, which means the number of ratings for Item I2 and Item I5 is not bigger than half of the total number of ratings. We can assume that Item I2 and Item I5 do not carry much weight with this matrix. Hence, when matrix has impactful items like Item I1, Item I3, and Item I4 as seen in [Table 4.8.2], running time of the recommendation system in computing similarity between items and to provide more accurate prediction is expected to reduce.

**Steps for Model Based Collaborative Filtering**

*Step-1 Generation of Utility Matrix:* We have collected the Movielens 100K dataset from which we got user's explicit rating information and list of movies. Then we have combined both rating and movie data frames to generate the Utility matrix as described in section [4.5]

*Step-2 Setting Up SVD:* To set up the SVD we have used Scipy and Numpy Python libraries because it lets us choose how many latent factors, we want to use to approximate the original ratings matrix (instead of having to truncate it after). We have calculated the three decomposed matrices U, $\Sigma$ and V by considering only top k-features. For this dataset we have chosen k= 50.

*Step-3 Generation of prediction matrix for all user:* We now have everything we need to make movie ratings predictions for every user. We can do it all at once by following the math and matrix multiply U, $\Sigma$ and $V^T$ back to get the rank k=50 approximation of our utility matrix. Note that utility matrix has original rating given by users and prediction matrix has rating predicted by the SVD model.

*Step-4 Generation of SVD recommendation function:* Now we have created a function to return the movies with the highest predicted rating that the specified user hasn't already rated. The input this function is the predicted matrix and utility matrix and a specific user_id for which movies are going to be recommended. We have designed this function such that it will return the top 10 best movies that user haven't rated and that aligned with the user's preferences

| | userId | movieId | rating | timestamp | title | genres |
|---|---|---|---|---|---|---|
| **25** | 150 | 1356 | 5.0 | 854203229 | Star Trek: First Contact (1996) | Action\|Adventure\|Sci-Fi\|Thriller |
| **5** | 150 | 32 | 5.0 | 854203071 | Twelve Monkeys (a.k.a. 12 Monkeys) (1995) | Mystery\|Sci-Fi\|Thriller |
| **12** | 150 | 141 | 5.0 | 854203072 | Birdcage, The (1996) | Comedy |
| **17** | 150 | 648 | 4.0 | 854203072 | Mission: Impossible (1996) | Action\|Adventure\|Mystery\|Thriller |
| **2** | 150 | 6 | 4.0 | 854203123 | Heat (1995) | Action\|Crime\|Thriller |
| **4** | 150 | 25 | 4.0 | 854203072 | Leaving Las Vegas (1995) | Drama\|Romance |
| **6** | 150 | 36 | 4.0 | 854203123 | Dead Man Walking (1995) | Crime\|Drama |
| **7** | 150 | 52 | 4.0 | 854203163 | Mighty Aphrodite (1995) | Comedy\|Drama\|Romance |
| **23** | 150 | 805 | 4.0 | 854203230 | Time to Kill, A (1996) | Drama\|Thriller |
| **20** | 150 | 780 | 4.0 | 854203071 | Independence Day (a.k.a. ID4) (1996) | Action\|Adventure\|Sci-Fi\|Thriller |
| **19** | 150 | 733 | 4.0 | 854203123 | Rock, The (1996) | Action\|Adventure\|Thriller |
| **15** | 150 | 608 | 4.0 | 854203123 | Fargo (1996) | Comedy\|Crime\|Drama\|Thriller |

Fig 4.8.1 List of explicit ratings given by the user_id 150 for corresponding movies

**Steps for SVD++ model**

As SVD++ is an extension of singular vector decomposition, some of the steps are similar.
*Step-1:* Generation of utility matrix
*Step-2:* Setting up SVD++ (same as SVD set up)
*Step-3:* Reduce the user rating matrix to binary matrix (1- high, 0-low)
*Step-4*: Generate the SVD++ recommendation function

```
predictions
```

| | movieId | title | genres |
|---|---|---|---|
| **574** | 736 | Twister (1996) | Action\|Adventure\|Romance\|Thriller |
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| **211** | 260 | Star Wars: Episode IV - A New Hope (1977) | Action\|Adventure\|Sci-Fi |
| **607** | 802 | Phenomenon (1996) | Drama\|Romance |
| **12** | 17 | Sense and Sensibility (1995) | Drama\|Romance |
| **87** | 112 | Rumble in the Bronx (Hont faan kui) (1995) | Action\|Adventure\|Comedy\|Crime |
| **558** | 708 | Truth About Cats & Dogs, The (1996) | Comedy\|Romance |
| **599** | 788 | Nutty Professor, The (1996) | Comedy\|Fantasy\|Romance\|Sci-Fi |
| **886** | 1210 | Star Wars: Episode VI - Return of the Jedi (1983) | Action\|Adventure\|Sci-Fi |
| **634** | 852 | Tin Cup (1996) | Comedy\|Drama\|Romance |

Fig 4.8.2 Movies recommended by model based collaborative filtering for a user_id 150

For instance, consider the fig. 4.8, which provides the list of explicit ratings given by a specific user with user_id 150 for the corresponding movies along with its genre. Fig 4.8.2. shows the output of the developed recommender system. That is, these are the movies recommended for user_id 150. These look like pretty good recommendations. It's good to see that, the truncated matrix factorization features "picked up" on the underlying tastes and preferences of the user. We have recommended some Action, Adventure, Romance, Thriller movies - all of which were genres of some of this user's top-rated movies.

## 4.9. IMPLEMENTATION OF HYBRID RECOMMENDATION MODEL

To overcome shortcomings of an individual model, we have developed a hybrid model wherein we stack two different recommendation models. As discussed in section [3.3.3] there are many ways to combine models but for this instance we wanted to choose a model with higher accuracy. After evaluating all the content-based and collaborative filtering models, we have found that the top two recommendation models are Content-Based Recommendation and SVD++ model-based recommendation, both having accuracy of 93.20% and 93.80% respectively. The proposed Hybrid model for our recommendation system is shown in Fig 4.9.1, below:


Fig 4.9.1 Hybrid Recommendation Model

The below flowchart shows the total workflow of the hybrid movie recommendation system that we have implemented.

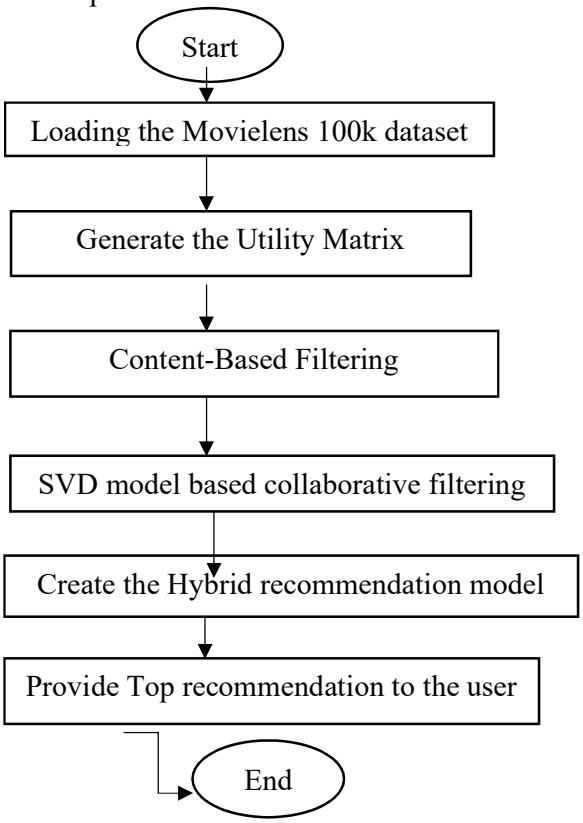



Fig 4.9.2. Flow Diagram of Implementation Hybrid movie recommendation system

# CHAPTER 5: PERFORMANCE EVALUATION

## 5.1. EVALUATION MEASURES

The purpose of evaluation in Machine Learning is to determine the usefulness and effectiveness of our learned classifiers (hypotheses) or of our learning algorithms on various collections of data sets. Optimally, the evaluation process would include the following careful steps:

1. Decide what "interesting" properties of the classifier should be measured, and choose an evaluation metric accordingly.

2. Decide what confidence estimation method should be used to validate our results.

3. Check that the assumptions made by the evaluation metric and the confidence estimation method are verified on the domain under consideration.

4. Run the evaluation method, using the chosen metric and confidence estimation method, and analyze its results.

5. Interpret these results with respect to the domain.

A straightforward method of measuring the recommendation quality is to measure the following:

- ***Mean Absolute Error (MAE)*:** this method simply takes the mean of the absolute difference between each prediction and rating for all held-out ratings of users in the test set.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

- ***Root mean squared error (RMSE)*:** is a related measure that has the effect of placing greater emphasis on large errors: on a 5-star scale, the algorithm is penalized more for being 2 stars off in a single prediction than for being off by 1/4 of a star 8 times. It is computed like MAE, but squares the error before summing it.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T} (\hat{y}_t - y_t)^2}{T}}$$

## 5.2. Evaluation of Content-Based Recommendation System using KNN

As the movie recommended by content-based filtering is based on genres, for evaluating model we have clustered movie based on groups of genres with the KNN classifier. The classifier label returned by KNN classifier to the movies recommended by the content-based filtering is compare to the classifier label of the movie on which model recommends, the hit and error are calculated accordingly to measure accuracy.

**Hit:** true count/total = **0.932**

**Fault**: false count/total = **0.067**

## 5.3. Evaluation of Collaborative Filtering

We have implemented collaborative filtering using both memory-based and model-based approach. Later on, those memory-based filtering that is user-user and item-item filtering are taken and passed to SVD model to predict rating of them for given user. These ratings are further taken to calculate the hit ratio. If rating predicted by model is more than 3 than the recommendation is hit.

$$HitRatio = \frac{recommendedMoviesRating > 3}{totalNumberOfRecommendedmovies}$$

**Hit ratio of model**
Hit ratio of User-user collaborative filtering: 0.7778
Hit ratio of Item-Item collaborative filtering: 0.8888

## 5.4. Evaluation of improved recommendation model

To build a robust recommender system, we have developed a model which factors in both explicit and implicit user feedback by incorporating SVD++ model with our existing recommendation model. Nonetheless, we have found that incorporating this kind of implicit data which inherently exist in every rating-based recommender system significantly improves prediction accuracy.

Both SVD and SVD++ models have been evaluated using RMSE (Root Mean Square Error). After fine tuning the important hyper parameters (number of factors, number of epochs, learning rate and regularization term) for SVD and SVD++, we got a RSME of **0.87** for SVD and **0.938** for SVD++.

## 5.5. Comparison of All Recommendation Techniques

All recommendation techniques implemented on our Movielens 100K dataset and their accuracy is listed on the table below.

| **Recommendation Techniques** | **Evaluation Measure** | **Model Accuracy** |
|---|---|---|
| Content based-filtering | KNN | 93.20% |
| User-user collaborative filtering | RMSE | 77.80% |
| Item-item collaborative filtering | RMSE | 88.90% |
| SVD model-based filtering | RMSE | 87.00% |
| SVD++ model-based filtering | RMSE | 93.80% |

Table 5.5.1. Comparison of Recommendation Techniques

From the above table, we can see the best two recommendation techniques that can be used to build the hybrid model are content-based filtering and SVD++ model-based

filtering with accuracy 93.20% and 93.80% respectively. Note that the accuracy values for these techniques can differ according to dataset used.

# CHAPTER 6: RESULT

In this project we have implemented and evaluated all the basic recommendation techniques. The SVD and SVD++ models are evaluated using RMSE, content based and collaborative based models are evaluated using hit ratio. In content-based filtering, we got model accuracy of 0.932. In Memory based collaborative approaches, hit ratio of user-user filtering is 0.778 and for Item-Item filtering is 0.889. In Model based collaborative approaches, the RMSE for SVD and SVD++ models are 0.87 and 0.938 respectively. We have observed that content-based filtering and SVD++ model-based collaborative filtering have provided greater accuracies. Thus, we have built the hybrid movie recommendation system by incorporating those two models as described in section [4.9].

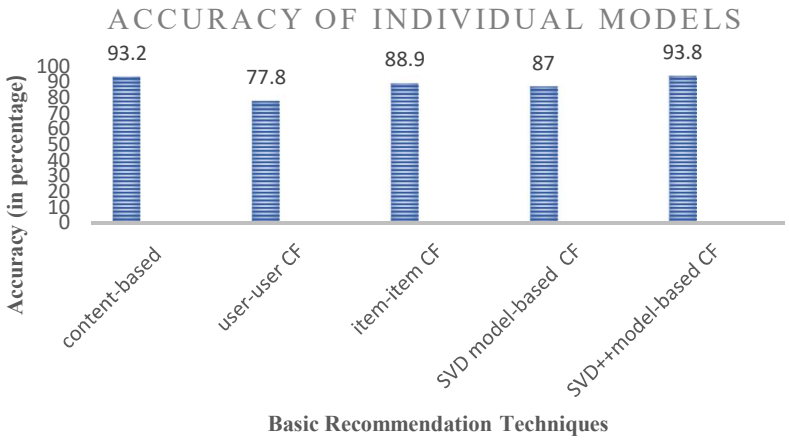



Fig 6.1 Accuracy of all basic recommendation techniques

# CHAPTER 7: CONCLUSION AND FUTURE WORK

Recommendation systems are increasingly gaining popularity due to their high number of applications. The users cannot manage all the information available on the internet. So, it is necessary for content providers to filter and customize the content shown to each user. Showing the right recommendations results in improved user engagement and in turn results in higher revenues. As per the algorithms discussed in this study, one recommendation system can have combination of algorithms as per different scenarios. So, if we have a metadata about the Movies that is genre, artists, production house, then, on basis of that we can recommend movies to user. User-user filtering suffers from the cold start problem. The system needs to wait until the user makes some purchases and rates them. Only then similar users can be found and recommendations can be made. However, Item-Item filtering overcomes this problem by finding similarity between items and recommending relevant items to users Further, SVD and SVD++ approach the problem by Matrix Factorization and dimensionality reduction. These algorithms provide good predictions for ratings a user might give to movies which he hasn't

watched. The Hybrid model we developed, filters the results further and provides accurate recommendation of movies.

## REFERENCES


[1]Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[2] Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010

[3] Costin-Gabriel Chiru, Vladimir-Nicolae Dinu , Ctlina Preda, Matei Macri ; "Movie Recommender System Using the User's Psychological Profile" in IEEE International Conference on ICCP, 2015.

[4]Google Recommendation system
https://developers.google.com/machinelearning/recommendation

[5] Luis M. de Campos, Juan M. Fernández-Luna *, Juan F. Huete, Miguel A. Rueda-Morales;"Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian Networks", International Journal of Approximate Reasoning, revised 2010.

[6] Urszula Kużelewska; "Clustering Algorithms in Hybrid Recommender System on MovieLens Data", Studies in Logic, Grammar and Rhetoric, 2014.

[6]WikiRecommenderSystem
https://en.wikipedia.org/wiki/Recommender_system#References

[7] https://www.datacamp.com/community/tutorials/recommender-systemspython

[8]https://towardsdatascience.com/various-implementations-ofcollaborative-filtering 100385c6dfe0

[9] Harpreet Kaur Virk, Er. Maninder Singh," Analysis and Design of Hybrid Online Movie Recommender System"International Journal of Innovations in Engineering and Technology (IJIET)Volume 5 Issue 2, April 2015.

[10] Manoj Kumar, D.KYadav, Ankur Singh, Vijay Kr. Gupta," A Movie Recommender System: MOVREC" International Journal of Computer Applications (0975 – 8887) Volume 124 – No.3, August 2015.

[11] https://github.com/gpfvic/IRR

[12] Utkarsh Gupta1 and Dr Nagamma Patil2," Recommender System Based on Hierarchical Clustering Algorithm Chameleon" 2015 IEEE International Advance Computing Conference (IACC).

[13] Collaborative Recommendations Using Item-to-Item Similarity Mappings Archived 2015-03-16 at the Wayback Machine.

[14]Intro to Recommender Systems: Collaborative Filtering; http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative -filtering/

[15] https://medium.com/@james aka yale/the-4-recommendation-enginesthat-can-predict-your-movie-tastes-bbec857b8223

[16] https://en.wikipedia.org/wiki/Singular value decomposition

[17] https://surprise.readthedocs.io/en/stable/index.html

[18] https://www.quora.com/Whats-the-difference-between-SVD-and-SVD++

[19] https://blog.statsbot.co/singular-value-decomposition-tutorial-52c695315254

[20] https://www.quora.com/Whats-the-difference-between-SVD-and-SVD++

[21] https://medium.com/recombee-blog/machine-learning-forrecommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed

[22] http://www.awesomestats.in/python-recommending-movies/

[23] https://ieeexplore.ieee.org/document/8058367

[24] http://recommender-systems.org/content-based-filtering/
[25] https://hackernoon.com/the-fastest-way-to-identify-keywords-in-newsarticles-tfidf-with-wikipedia-python-version-baf874d7eb16
[26] https://www.researchgate.net/publication/215470714