Movie Recommender System
Interim Report

DT282
BSc in Computer Science International

Mariana Pirtac

C17378303

Denis Manley

School of Computer Science

Technological University, Dublin

08/04/2022

Abstract

Throughout the past few decades, recommendations systems have been used on popular platforms. Without realizing it, we have all encountered a recommendation system. For example, when you go to YouTube, the site is recommending videos that you may find interesting. On Netflix you can see recommendations for movies you might enjoy. On Amazon, you can see recommendations for CDs, movies, and other products you might like. Most of the time, the recommendations you are given through these platforms are actually things you are interested in. That's a recommendation system working in the background and recommending things for you.

Many movies recommendation systems exist out there, but none of them are 100 percent accurate. This is why creating a recommender system remains a popular trend among software developers. A major goal of this project is to investigate why the existing movie recommendation systems may not be as accurate as they are believed to be and to develop an optimal movie recommendation system. This will be accomplished by using the Content-Based Recommender System approach(it will look at your profile and what are your area of interest and the genre of movie which you have previously chosen, in other words, it suggests similar movies to those that you have previously watched) and the Collaborative Recommender System approach(it looks at what other people with similar interests to yours, have watched if you haven't watched a movie which the other people similar to you have, that movie will be recommended to you). Furthermore, analysing data and applying machine learning will be a major part of this project.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Mariana Pirtac

_____

Mariana Pirtac

21/12/2021

Acknowledgements

I would like to express my sincere thanks and gratitude to my supervisor Denis Manley, for all the guidance, advices, support, and patience throughout this project.

Table of Contents

1. Introduction

1.1. Project Background

Nowadays thanks to the Internet there is a wide variety of choices. As a result, users had trouble choosing. In order to help users' choices, various organizations have developed recommendation systems. These systems help users choose what they want without having to deal with thousands of data information. These systems provide you with information based on the users' previous interests, behaviors, and user profiles. Instead of presenting the user with every possible option, they are given only what might be of interest to them. As a result, it benefits both the users and the businesses. This is because if a user is overwhelmed by the number of options they have to choose from, they will simply leave the website. However, since there are only a few choices available, there is a higher likelihood of a user using the given service or product.

Without realizing it, we use recommendation systems daily. Some of the most famous recommender systems are the ones used for YouTube, Amazon, and Netflix. If you go to one of the listed platforms, one of the things you will see on the main page will be Things you might like/Things you might want to Watch and then videos (YouTube) or products (Amazon) or movies (Netflix). That is the recommender system recommending you something you might like based on what you bought (Amazon) or watched before (YouTube, Netflix).

Although the idea of creating a recommendation system has been out there for the past few decades, and a lot of research has been done in this area, recommender systems are still popular to this day. One of the main reasons for that is that the recommender systems are not 100 percent accurate which means that recommender systems can be improved. All the present recommendation systems which I found; the accuracy achieved by the recommendation system developed has increased compared to the accuracy of the existing one. However, the accuracy is still not 100 percent. In chapter two I described some of the movie recommendation systems I was impressed by.

What is interesting about these recommendation systems is that when developing them they don't restrict the developer to using only one technique or only one algorithm. When developing a recommender system, the developer can use as many algorithms and as many techniques as they want. Incorporating more than one technique has proven to give a higher accuracy at least in the projects mentioned in chapter two.

One of the main reasons I choose to develop a movie recommendation system is because I am not restricted when choosing how to create this system. I could use one algorithm, or I could use three or more. The main goal of this project is to achieve high accuracy when getting results. For this project, supervised and unsupervised machine learning techniques will be used. Also, hybrid filtering will be incorporated. This filtering technique combines content-based filtering and collaborative filtering. Machine learning techniques, as well as the filtering methods, are described in chapter two.

1.2. Project Description

The purpose of this project is to develop an optimal movie recommender system that will provide better accuracy than the existing ones. The system will be developed using supervised and

unsupervised machine learning techniques to predict and recommend the movies the users might be interested in. I am also planning to use the content-Based Recommender System approach(it will look at your profile and what is your area of interest and the genre of movie which you have previously chosen, in other words, it recommends similar movies which you have watched in the past) and the Collaborative Recommender System approach(it looks at what other people who have similar interests as you have watched if you haven't watched a movie which the other people similar to you have, that movie will be recommended to you). The two approaches combined together form the Hybrid approach.

For this project, I will be using a dataset containing a list of the movies as well as features relevant to each movie. I am planning to start by working with a small data set and then increase the data set. Using a smaller data set will give me the opportunity to experiment with the algorithms. It will help me decide which algorithm is the most effective and see how several algorithms can be combined together. However, before implementing any of the algorithms, each dataset will be analysed thoroughly. Analysing the datasets is one of the major parts of this project.

In the end, I am planning to test it against the results gained from other existent movie recommender systems. If the accuracy of the movie recommendations system will be high, then I will develop it into a web app and I will ask a group of volunteers to try and test it.

1.3. Project Aims and Objectives

The main aim of this project is to develop an optimal movie recommendation system that begins with specific data sets and a number of ways of generating recommendations. The system should give accurate recommendations.

1.3.1. Literature Review

- The first objective is to analyse the principles behind existing recommender systems.
- The second objective is to analyse the methods commonly used in recommender systems and those specifically used in Film Recommender Systems.
- The third objective is to search for the most common data sets used as inputs in existing movie recommender systems. Then analyse the data and determine the most important features and their importance for the different recommendation system approaches.
- Investigate the main limitations of existing recommender systems and how they may be managed.
- Evaluate the most common methods used in determine the accuracy of recommender systems.
- Carry out user surveys on features they consider important in choosing a movie and what features they would like to include in their ideal recommender system including the amount of personal information they would freely give to improve recommendations.

1.3.2. Design

- Design an application that contains the most important recommender systems requirements found in the background research.

1.3.3. Implementation

- Implement an application based on the design and literature reviews.
- Then develop a software system, for a given data set, that allows the user to manipulate the data.
- Apply different ML techniques to the data set to produce a range of recommendations.

1.3.4. Test and Evaluation of the System

- Then analyse the efficiency of the system in terms across different subsets of data and different ML techniques.
- Produce a system based on this analysis that will produce a robust system with different approaches or a combination of approaches.
- Generalise the system so that it could be used for different types of data sets.
- Then incorporate other ML techniques developed by other students that have been generalised for a range of data sets.
- After that compare, the movie recommendation systems developed against existing movie recommendation systems and check the accuracy.
- Then incorporate the system into a web application.
- Find a number of volunteers to test the system created.
- Finally, analyse the outputs from both a technical perspective and a user perspective.

1.4. Project Scope

The scope of this project is to create an optimal movie recommendation system by combining various algorithms and machine learning techniques. This project is not about using every possible algorithm that there is. Instead, it is about finding the most appropriate algorithms that when combined together, will give accurate results. Also, this project is not about creating just another recommender system that already exists, instead, it is about developing a unique recommender system that will give high accuracy.

Apart from finding the most appropriate techniques, it also focuses on finding the most appropriate datasets. Currently, there are a large number of data sets available, containing both useful and redundant information. A major part of this project is to analyse existing datasets. This will enable us to be able to choose which data will be useful when creating a recommended system and which is redundant.

This project will be focusing mainly on the following areas of computer science: Machine Learning and Data Analytics.

1.5. Thesis Roadmap

Chapter 2: Literature Review

This chapter covers the literature research done for this project. In this chapter, some of the existing recommendation systems will be discussed. Also, it includes a summary of machine learning techniques and recommender systems methods. Furthermore, it includes an analysis of various datasets. Also, two previous final-year projects which cover a similar topic to the one used for this project were analyzed. Moreover, some of the technologies researched for this project were included.

Chapter 3: Prototype Design

This chapter presents a description of the methodologies used for this project as well as a system overview.

Chapter 4: Prototype Development

This chapter presents a description of the development process of the system.

Chapter 5: Testing

This chapter presents a description of the testing process of the system.

Chapter 6: Future Work

This chapter describes the plans for future development.

2. Literature Review

2.1. Introduction

In this chapter a review of relevant research is presented as it relates to the movie recommender system. First, existing movie recommender system will be presented. Following that, the technologies that might be presented will be listed and explained. Then, it will be explained what machine learning is and the types of machine learning. After that, it will be explained what a recommender system is and the types of recommender systems. Then the datasets used will be analysed. Finally, relevant final year projects will be discussed.

2.2. Recommendation System

Recommender systems are machine learning systems that help users discover new products and services. [13] These systems aim to predict users' interests and recommend product items that quite likely are interesting form them. [14] It can recommend a book which the user might like to read, a song the user might want to listen, a movie the user might want to watch and so on. In other words, recommendation systems are designed to help users have a more personalized experience on different platform. Without knowing we come across recommendation systems every day. Some of the most famous platforms used nowadays includes:

- Social media recommenders: Facebook, Instagram, Tiktok
- Retail: Amazon and eBay
- Music Spotify and YouTube
- Movie: Netflix and Amazon Prime

All of these platforms use recommendation systems which help the users to find products suitable or tailored to their tastes based on their previous viewing or buying habits. [13][14][15]

A recommendation system uses algorithm which suggests relevant products or movies (in this project) to the user, depending on the platform used. There are three main approaches used in recommender systems:

- Content Based Filtering Systems
- Collaborative Filtering Systems
- Hybrid Filtering Systems

Since this project is a movies recommender system the remainder of project will only focuses on examples related to movies as a discussion of a general recommender systems is beyond the scope of the dissertation.

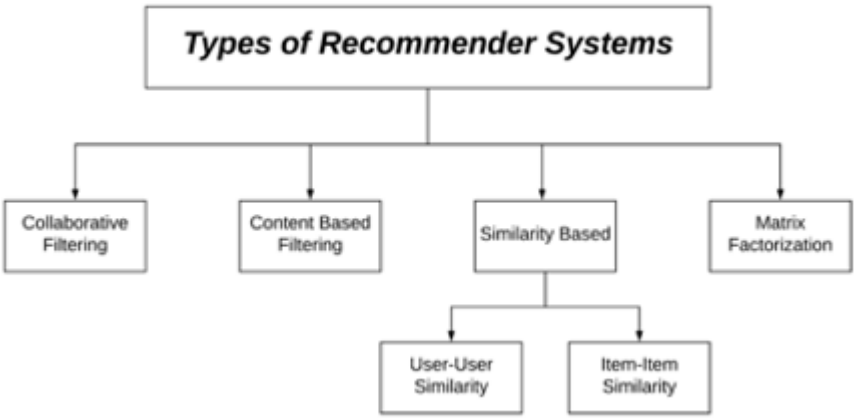
The three types of approaches differ by the type of movie data used. For content-based filtering systems, characteristic features of movies such as Genre, Actors, Directors original language will be used to analyse films. User's Gender, Age and Occupation will also be used to analyse the films.

For collaborative filtering, user item interactions information will be used. The Hybrid filtering combines the data used for content-based filtering and collaborative filtering.[14]

The advantages of using Recommendation systems are: it helps the users to find movies of interest, helps different companies to deliver their movies to the right user, helps websites to improve user-engagement, and it increases revenues for business through increased consumption. [14]

For this project a Movie Recommendation System will be built. In order to decide the type of filtering which will be used for this project, each approached be reviewed and analysed. The purpose of that is to find the most optimal way of designing and implementing a movie recommendation system.

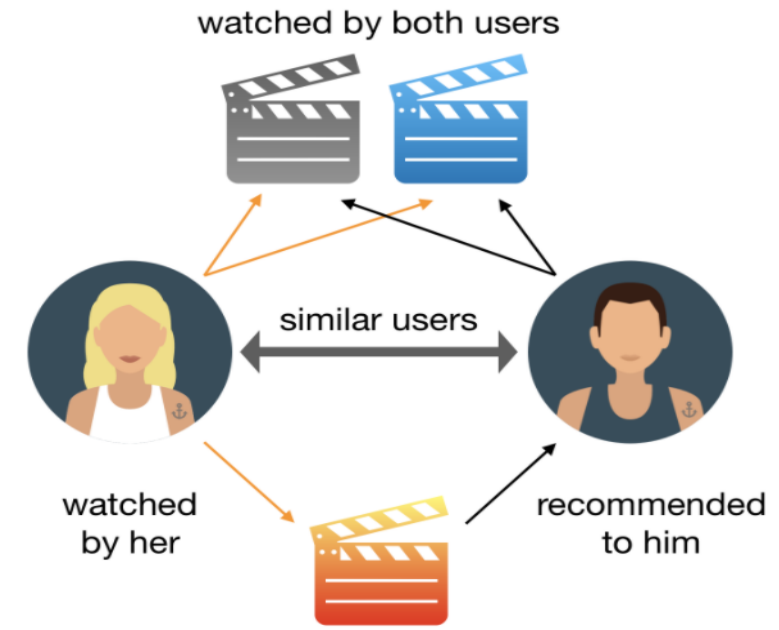
The main types of recommender system approaches as shown in **Error! Reference source n ot found.**.



[23] Figure 22. 1 The General Classes for Recommender Systems.

2.2.1. Collaborative Filtering

Collaborative filtering is one of the most frequently used approaches and usually provides better results than content-based recommendations. [14] Collaborative filtering focuses on the past interactions between users and items, to produce new recommendations as illustrated in [18] Figure 2. 1. These interactions are stored in a **user-item interaction matrix**. This interaction matrix is used to detect, e.g. movies, similar to ones previously viewed by the user. Then based on the findings it will be able to recommend the appropriate items. [14][16]



[18] Figure 2. 1 Collaborative Filtering

Collaborative filtering is divided into two categories:

- **model-based approach**
- **memory-based approach**

2.2.1.1. Memory-Based Approach

The memory-based approach identifies *one cluster of users* and then uses the interaction of one specific user to predict the interactions of other similar users.

Another way of using this approach is to identify clusters of items that have been rated by user A and use them to predict the interaction of user A with different but similar item B. [14] This approach deals with values of recorded interactions and are based on nearest neighbour search. [19] Shuyu Luo has used this approach in her paper. In this paper it is explained that there are two approaches:

- user - based collaborative filtering
- item - based collaborative filtering

There is an n * m matrix of rating, where n represent the user matrix and m represents the item matrix. A user can be represented by ui, i=1 and an item can be represented by pj, j=1. To predict the rating rij if the target did not watch the item j, we will have to calculate the similarities between target user I and all other users. To achieve this the top X similar users are selected and their weighted average of ratings where their similarities are represented by weights. There are three common ways to calculate the similarities:

- **Pearson Correlation**
- **Cosine Similarity**

- **Pearson Correlation**

2.2.1.2 Model-Based Approach

The Model based approach comes with a generative model which explains the user item interactions, then based on the model, information recommendations are made. The main goal of this methods is to train models to be able to make predictions. For example, using a user-item to train a model to predict the top 5 items that a user might like. [14] Figure 3. 1 A user item model associated with Collaborative Filtering Recommender System shows an example of how a user item model works. In the figure there are five users and five items (movies). The arrows represent which user has watched which movies. All five users have watched MOVIE-1 and MOVIE 2. USER-1, USER-2, USER-3, and USER-4 have also watched MOVIE-3, however USER-5 haven't watched it. So, MOVIE-3 can be recommended to USER-5. MOVIE-4 cannot be recommended because only USER-3 watched it. Also, MOVIE-5 cannot be recommended since only USER-1 watched it. This is a simple example of how collaborative filtering works.

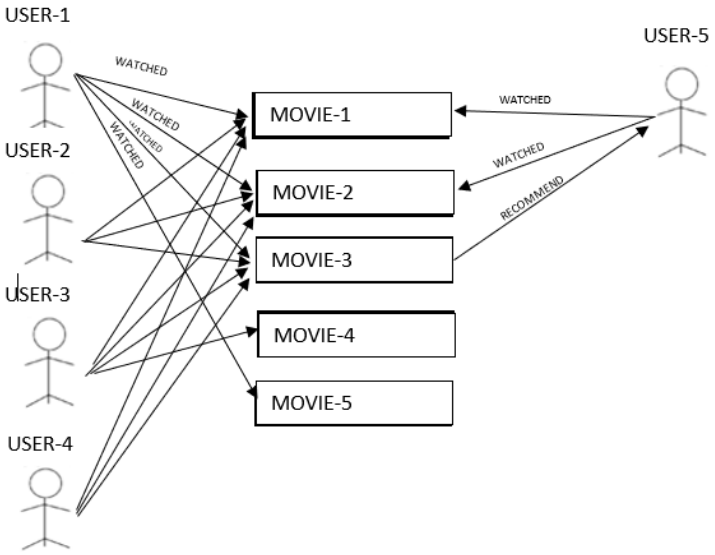


Figure 3. 1 A user item model associated with Collaborative Filtering Recommender System

2.2.1.3. Sparse Rating Matrix: Similarity

User based nearest neighbour attempts to predict the rating of a movie j that user I has not seen or rated: Firstly, you determine the similarities between the different users based on their rating history and then chose the users who are most similar. Calculate the weighted average rating for these users K as shown in [19] Equation 1. 1, where I is the user; j is the movie r is the rating and k is the set of similar users.

$$r_{ij} = \frac{\sum_{k} Similaries(u_i, u_k) r_{kj}}{number\ of\ ratings}$$

[19] Equation 1. 1 Calculate the rating of a movie

Similarity rating can be calculated using different distance measurements such as the **cosine similarity**, **Euclidean Distance** and **Pearson Correlation** which are described in the next section. shown in [19] Equation 1. 1 and illustrated in Figure 7. Item Matrix 1 and [20] Figure 8. 1**.**

Item based collaborative filtering finds item similarity and user ratings on items in a similar approach to the user-based collaborative filtering. However, since most movie rating datasets are very sparse it is better to use the matrix factorisation approach.[19] This method which predicts ratings in a user movie rating matrix overcomes the sparsity limitations and is discussed in detail in the next **section.**

2.2.1.4. Collaborative Filtering using User-Item Matrix Factorization

USER-ITEM Matrix

	I_1	I_2		I_j		I_{m-1}	I_m
U_1			...		...		
U_2			...		...		
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
U_i			...	A_{ij}	...		
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
U_{n-1}			...		...		
U_n			...		...		

[23] Figure 27. 1 User-Item Matrix: U1 (user 1, I1 is Item 1 (e.g. a movie) and Aij is the rating user I gives to item j

	Movie1	Movie2	Movie3	Movie4	Movie5
U1		5	4	2	1
U2	1			5	3
U3	1	4	4	1	
U4			2		2
U5	3	1	1		

[26] Figure 28. 1 A sparse user-item matrix

Most movie collaborative filtering methods try to predict the rating similarity between users. However, since most **user item matrices are quite sparse** it is best to use a method or to adopt a method which will predict ratings in the spare cells: a commonly used methods to deal with sparse user item rating matrix is Matrix Factorisation. Using matrix factorisation is more advantageous compared to other methods because even though two users haven't rated the same movie it is still possible to find the similarities between them if they share similar underlying tastes. This is done using latent features. The matrix contains how much a user is aligned with a set of latent features, and how much a movie fits into this set of latent features. An example of a latent feature is the genre (Action, Romance, Sci-Fi….) [19]

Singular value decomposition is a popular technique for collaborative filtering. It consists of decomposing the user-item matrix into three elements: the user feature eigenvectors, the feature-item eigenvectors, and a diagonal matrix of eigenvalues. Doing this is very advantageous because it helps to understand the users in terms of the latent features that they prefer, while items (movies) are understood in terms of the latent feature and the degree to which they are present. So, the distance metrics can be used to compare users directly to items and then generate the best match. [20]

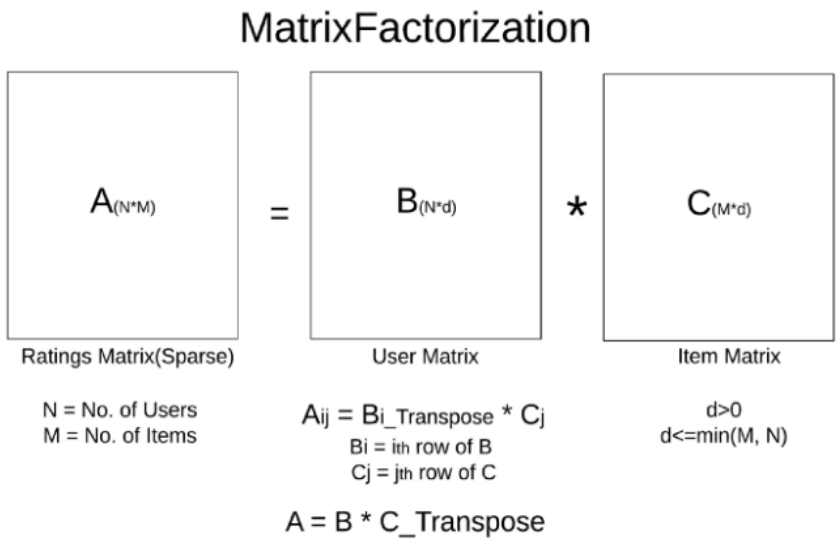


Figure 4. Matrix Factorization 1

In Figure 4. Matrix Factorization 1, A represents a user – item matrix, each cell in this matrix represents a rating given by the user to a movie(item). An example of such matrix(non-sparse) will look like this:

However, unless all users rate all the movies as shown in Figure 5. User Item Matrix 1, then it can result in a spare matrix. In the available data that rates movies such as the MovieLens 1 million dataset there are 6040 user, 3900 movies with on average each user rates 20 movies and the total number of ratings is 1000209 which gives a sparsity value of:

1 million MovieLens dataset sparsity:

- **1000209/3900*6040 = 0.042**

Matrix A can be decomposed into two matrices B and C. Matrix B represents the users while matrix C represents the items/movies. Since matrix A is achieved by multiplying N(the number of users) to M (the number of items) then matrix B(user matrix) is achieved by multiplying N(the number of users) to d and matrix C is achieved by multiplying M(number of item) to d. An example of matrix B and matrix C looks like this:

	Comedy	Action	Horror	Romance	Thriller
User 1	1	1	0	0	1
User 2	0	0	1	1	0
User 3	1	1	0	0	0
User 4	1	1	0	0	1
User 5	1	0	0	1	0

Fugure 6. User Matrix 1

	Comedy	Action	Horror	Romance	Thriller
Item/Movie 1	3	1	3	4	3
Item/Movie 2	1	2	1	3	1
Item/Movie 3	1	4	1	5	2
Item/Movie 4	3	1	3	4	2
Item/Movie 5	1	3	1	4	2

Figure 7. Item Matrix 1

In the figure we can see that A = B * C_Transpose, this means that A is a product of B and C_Transpose. Then Aij = Bi_Transpose *Cj, i represents the user and j represents the item. In other words, Aij represents a rating given by i (a user) to j (an item). Bi represents a row in the user matrix and Cj represents a row in the item matrix.

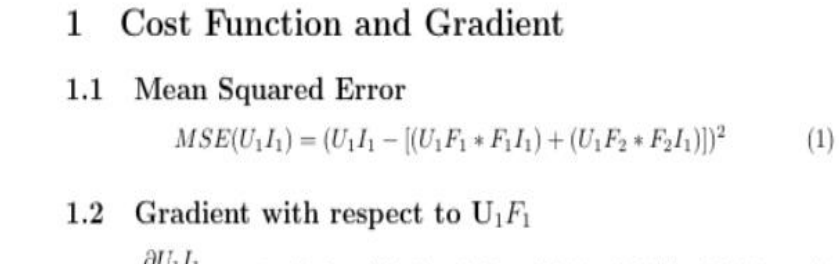
Another way of representing how matrix factorization works is shown in [20] Figure 8. 1. This figure represents how gradient descent-based matrix factorization works. The main thing is to create parameters and iteratively to update them.[20]

[20] Figure 8. 1 The dot product of a user-feature row and feature-item column is a single-item rating

In [20] Figure 8. 1 each cell in both the user-feature matrix and the feature-item matrix is a parameter. We need to update these parameters iteratively through some cost function. Before doing that, we need to know the user-item cells that any given user-feature cell and feature-item cell contribute to. For example, U1F1 and U1F2 contribute to U1I1, U1I2, and U1I3. The matrix multiplication is executed as the dot product of the row from the left matrix with the column of the right matrix. The row and the column indices of the user-item matrix determines the whole row and column selected from the decomposed matrices. [20]

2.2.1.4. Cost Function and Gradient Descent

To determine values in the user and item feature matrices the system uses a cost function to determine the mean squared error and back propagation gradient descent to reduce the error. This is a similar approach to neural networks, what it does is, it modifies the values in the item and user matrix. When an optimal value is reached the values in the user-item matrix are obtained by multiplying both user-feature matrix and feature-item matrix as illustrated in the [20] Figure 8. 1. This is then used in the collaborative approach to select or suggest films to user by determining the best rating in their rating vector. [20][19]



1 Cost Function and Gradient

1.1 Mean Squared Error

$$MSE(U_1I_1) = (U_1I_1 - [(U_1F_1 * F_1I_1) + (U_1F_2 * F_2I_1)])^2 \quad (1)$$

1.2 Gradient with respect to U_1F_1

$$\frac{\partial U_1I_1}{}$$

[20] Figure 9. Cost Function and Gradient (MSE of U1I1 and gradient with respect to U1F1) 1

In [20] Figure 9. Cost Function and Gradient (MSE of U1I1 and gradient with respect to U1F1) 1 we can see an example of how one (U1I1) of the three gradients necessary for U1F1 can be updated, (the other two gradients U1I2 and U1I3 should also be updated with respect to U1F1). When we update U1F1 we will average all three of these gradients.

The new U1F1 value is obtained by adding the product of the (average gradient for U1F1 and a learning rate) to the current U1F1. In a sparse matrix where a user has not rated a movie the value of a cell in the user- item rating matrix may initially contain no value and is set to 0.

New U1F1 = learningRate*AvergageGradient(U1F1) + U1F1 (current)

This must be done for all cells in the User-feature and the Item Feature Matrices. The product of the user-feature matrix and the item-feature matrix (transpose) results in a predicted User-Item Matrix:

Pred_Matrix	M1	M2	M3	M4
U1	4.9921822	2.94665346	3.40536863	1.00099803
U2	3.97665318	2.15037496	3.10801712	0.99668824
U3	1.04690221	0.89961819	5.20187891	4.96401721
U4	0.98735764	0.77053916	4.27454201	3.97482436
U5	1.86869693	1.09103752	4.94375052	4.02158161
U6	1.93781997	1.06363722	3.01008954	1.99265014

[26] Figure 25. 1 Sample Predicted Rating after matrix factorization.

Original Matrix	M1	M2	M3	M4
U1	5	3	0	1
U2	4	0	0	1
U3	1	1	0	5
U4	1	0	0	4
U5	0	1	5	4
U6	2	1	3	0

[26] Figure 26. 1 Initial sparse matric before matrix factorization

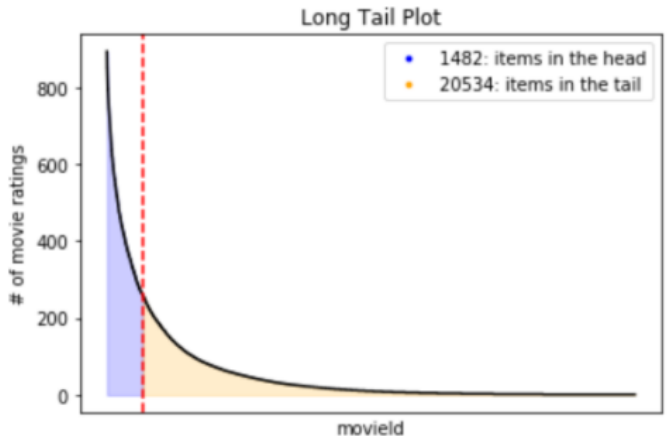
2.2.1.5. Collaborative Filtering Advantages and Disadvantages

Advantages:

- It doesn't require information about the users or items therefore it can be used in different situations.
- The more users interact with items, then the new recommendations become more accurate.
- There are numerous datasets available for use from movielens and other sources that have movie film rating data, such as the 1 million movielens data set.

Disadvantages:

- It can suffer from the "**cold start problem**" because it uses past interactions to make recommendations. The cold start problem occurs because it is impossible to recommend:
  - An item/movie to new users
  - A new item/movie to any users or items have too few interactions to be efficiently handled.
- The Long Tail problem: some movies have many ratings while a large number have only a few ratings as illustrated in [25] Figure 18. 1 :



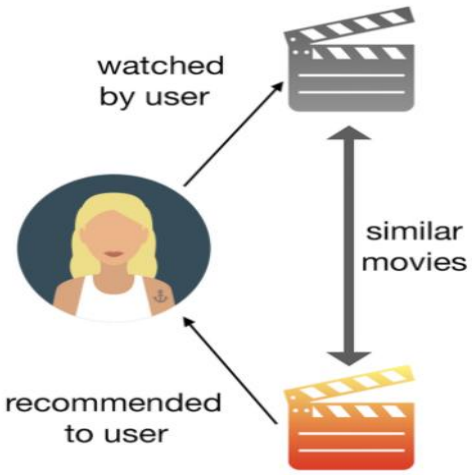
[25] Figure 18. 1 An illustration of the long tail problem which is the result of popularity bias.

2.2.2. Content Based Filtering

Content based filtering uses information about users and items. The main goal of this approach is to try to build a model, based on the available features, previous actions, or explicit feedback. The recent progress of pattern recognition in machine learning unlocked great improvement in the content-based model using information extracted from raw images or raw text description.[14][16][17]

In content-based filtering users are recommended movies with similar content to those they have already viewed. An example of this is illustrated in [21] Figure 10. 1**.** When recommending movies to the user using this technique, it tries to guess the features or behaviours of a user, given the item's features which the user reacted positively to as shown in [22] Figure 11. 1. In [22] Figure 11. 1, the last two column are Action, and Comedy, which describes the genres of the movies. Given these genres, we can know which users likes which genre. Due to this we can obtain features corresponding to that particular user, depending on how the user reacts to movie of that genre. [22]

When creating a content-based filtering matrix we can populate it with information about the user and information about the movies. We can use the following type of information about the user: Gender, Age, and Occupation. For the information about the movie, we can use: Genre, Actors, and Popularity.



[21] Figure 10. 1 Content Based Filtering

Movies	User 1	User 2	User 3	User 4	Action	Comedy
Item 1	1		4	5	Yes	No
Item 2	5	4	1	2	No	Yes
Item 3	4	4		3	Yes	Yes
Item 4	2	2	4	4	No	Yes

[22] Figure 11. 1 Content Based Filtering Table Example

2.2.2.1. Methods Used in Content Based Filtering

2.2.2.1.1. Item Content Domain

Many content-based recommender system use keywords such as genre, lead actor/actress, director, original language, year of release to build a recommender system as such data relates only to the movie and not the user.

2.2.2.1.2. User Content Domain / User Demographic

In content-based filtering, user content domain are often used. These are features that are related to users only, this includes: gender, age group, occupation, likes and dislikes of users.

2.2.2.2. Advantages and Disadvantages of using Content Based Filtering

Advantages:

- It avoids the "cold start problem".
- Content-based filtering doesn't require large dataset and expensive servers to train the recommendation engine.
- The model can capture the specific interests of a user and can recommend niche items that very few other users are interested in.

Disadvantages:

- It requires a lot of domain knowledge because the feature representation of the items is hand engineered, therefore the model can only be as good as a hand – engineered feature. It is hoped to design this using training sets to pick the features that are most relevant to user choice and then build the model on test data set.
- The model can only make recommendations based on existing interests of the user, therefore the model has limited ability to expand on the users' existing interests.

2.2.3. Hybrid Based Filtering

The Hybrid model is the combination of the collaborative filtering and content filtering. The combination can be made in two ways: we can either train two models independently and combine their suggestions or directly build a single model that unify both approaches. [17]

The advantage of the Hybrid Model is that is combines two recommendation techniques to gain performance. [16]

The disadvantage of the Hybrid Model is that if two recommendation techniques will be combined both will require a database of ratings.

The recommender system that it going to be implemented here is predominately a collaborative approach but will incorporate similarity matrices and other approaches that could be classes as content based. In this case the user will be able to select numerous ways to generate their movie list so at a certain level it will be a user driven Hybrid Recommender system.

2.3. Similarity Matrices: User: User and Item: Item

2.3.1. Data analysis for constructing User Vector

Based on the problem of cold start where in general there is limited information about the users and their preferences it must be decided to implement a demographic collaborative filtering as one approach to movie recommendations.

To implement such approach users will need to have up to three characteristics associated with them: their gender, their age, and their occupation. Therefore, it will produce individual user groups:

Number of genders * number of age groups * number of occupations

In the case of movielens 1 million data set there will be **2*7*20 = 280 groups**. Total users = 6040: average number per group about 20.

2.3.2. User-User Similarity Matrix

To construct this matrix a feature set relating to users must first be generated. This set can be based on numerous features. Using this feature set or feature vector numerous similarity measurements can be used. In [23] Figure 12. 1, we will find the similarities between users based on the ratings given by users. So, two users will be similar based on the similar ratings given by both of them. If any two users are similar, then it means that both of them have given very similar ratings to the items. The similarity between users will range from 0 to 1, where 1 means the highest similarity and 0 means no similarity. All the diagonal elements are 1 because the similarity of the user with himself is the highest. Sim12 is a similarity score of User U1 and User U2. Simij is a similarity score of user Ui and Uj. [23]

USER-USER SIMILARITY MATRIX

	U_1	U_2		U_j		U_{n-1}	U_n
U_1	1	Sim_{12}	...	Sim_{1j}	...		
U_2		1	...		...		
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
U_i		Sim_{i2}	...	Sim_{ij}	...		
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
U_{n-1}			...		...	1	
U_n			...		...		1

[23] Figure 12. 1 A sample user-user similarity matrix. A measurement of similarity (Sim) is in each element, and it shows the amount of similarity between two users.

2.3.2.1. User Feature Vector

The user feature vector which is used to determine the similarity can be based on numerous user features such as genre, rating, and user demographics (Gender, Age, and Occupation). In Figure 13a. 1 and Figure 13b. 1 we can see example of user demographic vector.

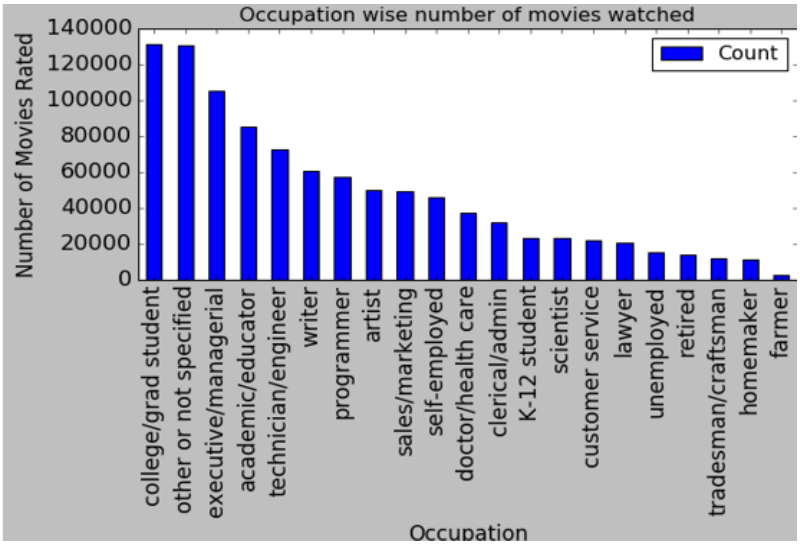


Figure 13a. 1 An example of a User Demographic Vector

User ID	Male	Female	Under 18	25-34	35+	Student	academic	Other
1	0	1	0	1	0	0	1	0
2	0	1	1	0	0	1	0	0

Figure 13b. 1 A sample user demographic feature vector used to construct a user-user similarity matrix.

2.3.2.2. User similarity based on Genres

A user movie rating vector would contain 1 for each movie a user has seen and a 0 for all movies it hasn't seen. A genre preference similarity vector could represent users based on their Genres preferences. As shown in Figure 14. 1. It is also possible to develop a user of all the movies and the rating value assigned to each but given the degree of sparsity of such vectors they will not be considered any further in the project.

User ID	action	adventure	animation	children	comedy	crime
1	0	0	0	1	1	0
2	1	0	1	0	1	1

Figure 14. 1 A sample user genre vector. 0 indicates that the user has not rated the movie and 1 indicates that the user has rated the movie.

2.3.3 Item-Item Matrix

An Item-to-Item matrix as with a user-to-user matrix requires that a feature set or vector for each movie is generate. This can be based on many features but to avoid complexity and given the fact that genre is arguably the most important feature only an item vector of the genre of a movie will be generate in this project. Most dataset have associated genres.

In [23] Figure 15. 1**,** we can see what the item-item similarity matrix looks like. Two items will be similar based on the similar ratings given to both items by all users. If any two items are similar, then it means that both were given a very similar rating by all the users. The similarities range from 1 to 0, value one means highest similarity, while value 0 means lowest similarity. All the diagonal elements will b 1 because an item has the highest similarity with itself. Simij represent a similarity score of user Ii and user Ij.

Using a vector with the same form as discussed for a user vector based on genre history each movie will have its own genre vector. An Item-to-Item cosine similarity and distance similarity are implemented in the **cosine_ similarity.py** shown later.

ITEM-ITEM SIMILARITY MATRIX

	I_1	I_2		I_j		I_{m-1}	I_m
I_1	1	Sim_{12}	...	Sim_{1j}	...		
I_2		1	...		...		
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
I_i		Simi2	...	Sim_{ij}	...		
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
I_{m-1}			...		...	1	
I_m			...		...		1

[23] Figure 15. 1 A sample item-item similarity matrix. A measurement of similarity (Sim) is in each element, and it shows the amount of similarity between two users.

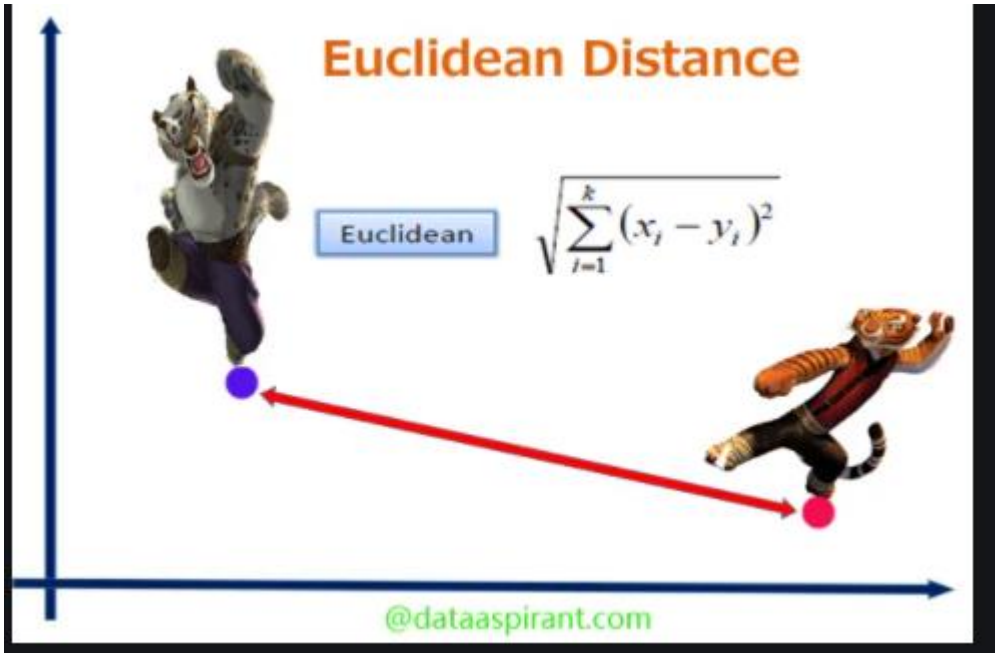
2.4. Measurement of Similarity

There are several methods that can be used to determine the similarity between two entities such as a movie or a user. Cosine similarity and Euclidean Distance use a vector to represent either an item or a user. A user vector might include features such as: Gender, Age, Occupation. While the item vector might include movie rating or genre preference.

Using these vectors, we can get a measurement score between the vectors for each user/item. The following two illustrate both measurements using 2 dimensions or Object Features.

2.4.1. The Euclidean Distance

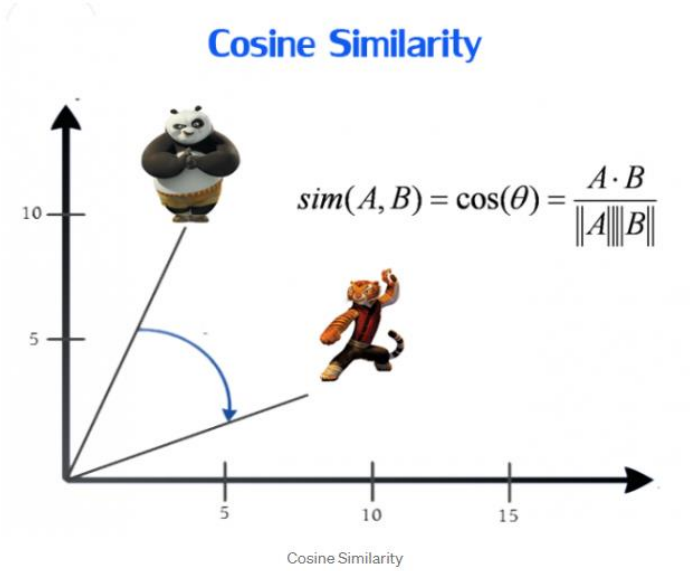
The Euclidian Distance also known as the mean squared distance shows the distance in feature dimensional space between the two objects [24] Figure 16. 1 shows how to calculate the distance in a two-dimensional feature space. However, it can be generalised to have any number of features, for example in the case of measuring genre similarity the feature space would have around 20 dimensions.



[24] Figure 16. 1 Euclidean Distance Similarity Measurement

2.4.2. Cosine Similarity

The cosine similarity is the cosine of the angle between the two objects in their feature space and is 0 if there is no similarity and 1 if there is 100% similarity. In other words the larger the value of the cosine the greater the similarity. [23] Figure 17. 1 illustrate the concept in 2-D feature space but it can also be generalise to any number of features.



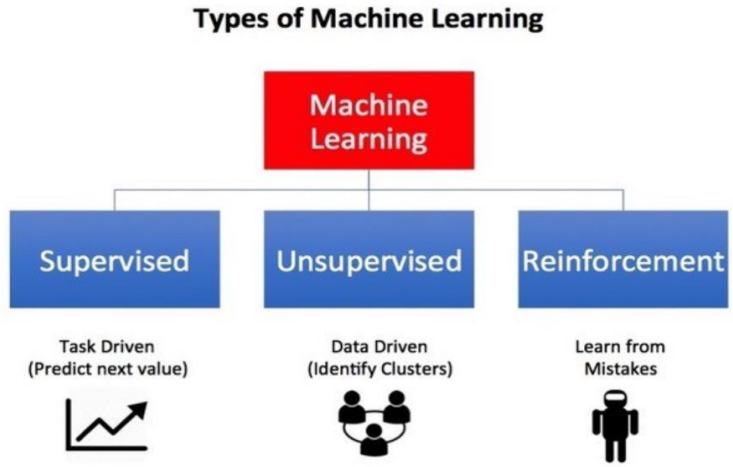
[23] Figure 17. 1 Cosine Similarity Measurement

2.5. Machine Learning

Machine learning is a subset of AI (artificial intelligence). It focuses on building applications which will learn from data and will improve the accuracy over time without being programmed to do so. This is achieved by using various algorithms. The algorithms are trained to search for common features and patters in large amount of data.[5] Building a machine learning model consists of four steps:

- Step 1: Selection and preparation of a training data
- Step 2: Choosing the algorithm
- Step 3: Training the algorithm to create the model
- Step 4: Using and improving the model

There are three main machine learning techniques: *Supervised, Unsupervised and Reinforced*. [6] Figure 1. 1 displays the three main machine learning techniques. To decide which machine learning technique will be best suited for this project in-depth research has been done.



[6] Figure 1. 1 Types of Machine Learning Techniques

2.5.1. Supervised Learning

Supervised learning is the type of machine learning technique which takes a known set of inputs data and known responses to the data and trains a model which will generate predictions for the

responses to the new data. [7] It is using the following formula "Y=f(x)". x represents the input variables while Y represents the output variables. From this formula we can see that the output(Y) is a dependent variable of input (x). In order to learn the mapping function from x to Y an algorithm will be used. The main goal of this technique is to approximate the best mapping function (f) which will be able to understand how the input should be matched with the output.[8]

Supervised Learning can be grouped into two groups regression problems and classification problems [9].

A regression problem is when the output is a real value, such as:

- "euro"
- "height"
- "weight"

A classification problem is when the output is a category:

- Like or Dislike
- Genre
- Rating where rating can be 1, 2, 3, 4 or 5

The algorithms commonly used for regression techniques are: **linear model**, nonlinear model, regularization, stepwise regression, **decision trees, neural networks**, and adaptive neuro-fuzzy learning. [7]

Since the output of recommender system are categories, classification can be best described as classification The algorithms commonly used for classification techniques are: SVM (support vector machine), **k-nearest neighbor**, **Naïve Bayes**, **discriminant analysis**, **logic regression**, **neural networks** and **decision trees.** [7]

2.5.2. Unsupervised Learning

Unsupervised Learning is a type of machine learning technique in which the model will discover patterns and information that was previously undetected. Usually, it deals with unlabelled data. This technique has only input data (x) and no corresponding output. Thus, unsupervised learning techniques allow the user to perform more complex processing tasks compared to supervised learning technique. It allows to find unknown patterns in data, features which can be helpful for categorization. Also, it is easier to get unlabelled data from a computer than labelled data.[10][9]

Unsupervised learning can be grouped into clustering problems and association problems. For clustering problems, the main goal is to discover the inherent grouping in the data, such as segmenting people with similar taste in movies. For association rule the goal is to discover the rule that describes large portions of data, an example is :

If a user(s) watch movie X they also tend to watch Y. [9]

Unsupervised learning uses more frequently clustering algorithms.[11] The clustering algorithms will find clusters/groups, but you can also modify the number of clusters in the algorithm. There are various types of clusters: exclusive clusters, agglomerate clusters, overlapping clusters, probalistic

clusters. The algorithms used for clustering are: **Hierarchical clustering, K-means clustering, K-NN(k nearest neighbors)**, principal component analysis, singular value decomposition, **Hidden Markov Model, Neural Networks** and independent component analysis.[7][10]

Association algorithm allows us to discover interesting relationships between variables in large databases. [10]

2.5.3. Reinforced Learning

Reinforced learning is a machine learning technique that trains models to make a sequence of decision in a complex environment. This technique uses an agent which makes the decisions based on the decision made the agent will either get a reward or a punishment. The goal of the agent is to get as many rewards as possible. Although the designer of the technique sets weather the agent gets a reward or not, the agent is not being given any hints on how to come up with the right decision. So, it is up to the agent to figure out how to come up with the right decision. So, the more decisions the agent is given the better it will get at coming up with the right decision. This machine learning technique learns on its own how to perform, while supervised and unsupervised learning are told how to perform. [12][11]

This technique requires a large amount of data. Therefore, it is important to use it only when large amount of data is provided (e.g. games). [12] [11]

Reenforced Learning uses the following algorithms: Monte Carlo, Q-Learning, State Action Reward State Action (SARSA), Deep Q Network (DQN), Asynchronous Actor-Critic Agent(A3C), Deep Deterministic Policy Gradient (DDPG), and NAF.[11]

2.5.4. Conclusion

After learning about each machine learning technique. It is clear that each technique is powerful and useful on its own. Each technique uses various algorithms which makes them unique. For my project I could use supervised and unsupervised learning. However, I shouldn't use reenforced learning because this technique is usual used in games or robotics.

2.6. Recommender System Issues

Collaborative Filtering recommender systems by its nature needs a significant amount of data related to anything new that is added to the system be it an item or a Movie or a User. In both cases the systems need user item interaction data. This is referred to as the cold start problem. A secondary but related issue is the so-called long tail problem both of these and other issues related to Recommender Systems be examined in this section.

2.6.1. Cold Start Problem

The cold start problem affects both a new user and a new item so, if possible, some taught needs to be consider to see if it may be possible to elevate the Cold start problem to some degree. The following considers what attributes need to be recorded for any new user or item to the system.

2.6.1.1. Adding a new user

When a new user registers with the system very little is known about them or their rating habits. Since we have no existing data, it is not possible to see their similarity to other users who have similar demographic characteristics such as:

- Gender
- Age
- Occupation
- Location

Nor is there any detail on their ratings history which does not allow the system to align them with similar users and make recommendations on movies these users most similar may have seen but the current user has or may not have seen. This user movie ratings could be based on: Actual movies and Genre of movies.

This however raises two major questions. If a user is registered:

- What data should be stored to keep track of their rating history
- What about ensuring the registration process complies with the GDPR regulations.

The GDPR issue will be resolved by encrypting the stored data and ensuring any not sharing personal user details with other users. This still allows us to use their rating history based on genre or actual movies to make recommendations to other similar users.

Ideally the following user data would prove useful. It was decided that the user data that will need to be stored includes

User Data:

- User Id
- User Name (visible only to user)
- Gender,
- Age,
- Occupation
- User Group ID (require for demographic recommender system)
- Main Genre Preferences
- List of movies already viewed

However, it will be necessary to carry out surveys to determine if user would be willing to share such data. This data collection feature will be built into the system when a user registers to use the application and probably as an optional but preferred feature and will be implemented using an opt-in option.

In addition, to improve the personalisation of the system the rating history of each user will also be recorded.

User Rating history:

- Movie Id
- Rating

2.6.1.2. Adding a new Movie

The same problem applies to movies that have not been rated.

When a new item coming in, until it has to be rated by substantial number of users, the model is not able to properly recommendation the movies recommended must at least have some associated ratings.

When a new movie is added ideally, we should add as much detail as possible but given the 1 million movielens datasets that forms the foundation of our recommender system the system will only store:

- MovieLens ID (if available)
- IMDB/TMDB ID (if available)
- Movie Name
- Genres
- Year of release
- Original Language
- Critically acclimated (yes/no)

Clearly only a system administrator will be given the option to add new movies to the system. The new movie ID will also need to be added to a rating data file so its average rating and rating count and who rated the new movie can be stored.

Similarly, for items from the tail that didn't get too much data, the model tends to give less weight on them and have popularity bias by recommending more popular items.

To limit this problem a content filtering approach will be applied. Firstly, as the user logs on or registers to use the system they will be asked to enter: their gender, age and occupation or some subset of them. If they do not want to add any details, they will be informed the system will not be adaptable as there is insufficient user data.

2.6.2. The Long Tail Problem

Another problem with collaborative systems is the long tail problem. Many datasets or data in recommender systems have an inherent bias as illustrated in [25] Figure 18. 1.

This is due to popularity bias and will often mean that many films even if they have very high ratings will not be recommended when using metric based on weighted ratings, as the popularity of a film will give it a biased advantage in a ranked list. In the proposed recommender system, the problem will be dealt with an option by removing the most popular films. [25] Figure 18. 1 illustrates the problem very clearly, It shows than the vast majority of films have a low rating count, in this illustration ratings of 200 or less and the rating count number drops very quickly. It is not true to say that over half the movies have a low rating count of 150 or less.

However, it must be stressed as it is evident from the table in [25] Figure 19. 1 that even if the film has a low count, it does not mean it has a low rating.

This long tail problem if completely ignored would be unable to recommend good films that are: rarely viewed or data related to them is not in the currently available datasets and may not suit users who are more “discerning”.

This can be used in a few ways. The most obvious is to recommend movies based on a film chosen by the user. However as described by Longo 2018 it may also be useful to give the user the option if they are tired of the “most popular films”. The system would then remove the top most popularity movies they system would find movies in this data set similar to the one chosen by the user.[25]

	Title	Genre	Count	Rating
0	Usual Suspects, The (1995)	Crime|Thriller	1783	4.517106
1	Lamerica (1994)	Drama	8	4.750000
2	Shawshank Redemption, The (1994)	Drama	2227	4.554558
3	Schindler's List (1993)	Drama|War	2304	4.510417
4	Close Shave, A (1995)	Animation|Comedy|Thriller	657	4.520548
5	Gate of Heavenly Peace, The (1995)	Documentary	3	5.000000
6	Godfather, The (1972)	Action|Crime|Drama	2223	4.524966
7	Schlafes Bruder (Brother of Sleep) (1995)	Drama	1	5.000000
8	Wrong Trousers, The (1993)	Animation|Comedy	882	4.507937
9	Follow the Bitch (1998)	Comedy	1	5.000000
10	Seven Samurai (The Magnificent Seven) (Shichin...	Action|Drama	628	4.560510
11	Apple, The (Sib) (1998)	Drama	9	4.666667
12	Sanjuro (1962)	Action|Adventure	69	4.608696
13	Ulysses (Ulisse) (1954)	Adventure	1	5.000000
14	Smashing Time (1967)	Comedy	2	5.000000
15	I Am Cuba (Soy Cuba/Ya Kuba) (1964)	Drama	5	4.800000
16	Baby, The (1973)	Horror	1	5.000000
17	Song of Freedom (1936)	Drama	1	5.000000
18	One Little Indian (1973)	Comedy|Drama|Western	1	5.000000
19	Lured (1947)	Crime	1	5.000000
20	Bittersweet Motel (2000)	Documentary	1	5.000000

[25] Figure 19. 1 The list of films with high average ratings from 1 million movielens dataset

2.7. Data Sets

Introduction in the previous sections the different recommender systems were discussed and based on it, certain data was considered essential to build them. These included user rating data for collaborative system; user data and item data for content-based system. The following section discuss the main datasets available for movie recommender systems and what each set contains.

2.7.1. MovieLens Data Sets

2.7.1.1. The small latest movielens dataset

It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996, and September 24, 2018. This dataset was generated on

September 26, 2018. Given the fact it is quite recent and relatively small it may be used to develop a proof of concept for a collaborative recommender system.

4 CSV files:

- links.csv
- movies.csv
- ratings.csv
- tags.csv

links.csv contains:

- movieId
- imdbId
- tmdbId
- **9743 rows**

movies.csv contains:

- movieId
- title
- genres
- **9743 rows**

ratings.csv contains:

- userId
- movieId
- rating (1 to 5)
- timestamp
- 100837 rows

tags.csv contains:

- userId
- movieId
- tag
- timestamp
- **3684 rows**

2.7.1.2. 25 million movielens data set

This dataset 25000095 ratings and 1093360 tag applications across 62423 movies. These data were created by 162541 users between January 09, 1995, and November 21, 2019 (movielens .org). Each user rated at least 20 movies. This dataset is very large and will not be used in the development of the recommender system.

2.7.1.3. 1 million Movielens Data Set

These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

3 .dat files:

- movies.dat
- ratings.dat
- users.dat

movies.dat contains:

- movie title and the year in which it was released
- movie genre (Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western)

ratings.dat contains:

- user ID (1 to 6040)
- movie ID (1 to 3952)
- ratings (1 to 5)
- timestamp

users.dat contains

- User ID
- gender (m or f)
- Age (represented by the following numbers: **1** (Under 18), **18** (18-24), **25** (25-34), **35** (35-44), **45** (45-49), **50** (50-55), **56** (56+))
- Occupation (represented by a number from 0 to 20:
  - **0** → other or not specified
  - **1** → academic/ educator
  - **2** → artist
  - **3** → clerical/admin
  - **4** → college/grad student
  - **5** → customer service
  - **6** → doctor/ health care
  - **7** → executive/ managerial
  - **8** → farmer
  - **9** → homemaker
  - **10** → K-12 student
  - **11** → lawyer
  - **12** → programmer
  - **13** → retired
  - **14** → sales/marketing
  - **15** → scientist

- **16** → self-employed
- **17** → technician/engineer
- **18** → tradesman/ craftsman
- **19** →unemployed
- **20** → writer)

- Zip-code

Changes made to the .dat files:

- The files are converted to CSV files
- In the users file the occupation will be represented by the name of the occupation instead of a number form 0-20.
- In the user file the age will be represented by the age group (e.g. under 18, 18-24, 25-34….) instead of just a digit (e.g. 1, 18, 25….)

2.7.2. IMDB dataset

5 CSV files:

- credits.csv
- keywords.csv
- links.csv
- rating.csv
- movies_metadata.csv

credits.csv contains:

- cast
- crew
- id (movie Id)
- **45505 rows**

keywords.csv contains:

- id (movie ID)
- keywords (related to movies)
- **461257 rows**

links.csv contains: (connect Movielens to

- movieId
- imdbId
- tmdbId
- **45844 rows**

rating.csv contains:

- userId
- movieId
- rating (1 to 5)
- timestamp
- **1048576 rows**

movies_metadata.csv contains:

- adult (this column is FALSE for every movie, meaning there are no adult only movies)
- belongs_to_collection
- budget
- genres (each movie has 1 or more than one genres)
- homepage (original homepage of each movie)
- id (movie id)
- imdb_id
- original_language
- original_title
- overview (brief description of the movie)
- popularity(Popularity is a value that usually is updated/calculated daily when calculating it, the following things are taken into account: the site views, number of user ratings/watchlist/favourite additions, and release date.)
- poster_path (link for movie posters)
- production_companies
- production_countries
- release_date
- revenue
- runtime
- spoken_languages (all the languages used in the movie)
- status (Released for every movie)
- tagline (Memorable phrase from the movie or that describes the movie)
- title (English titles)
- video (False for every movie, since there are no links for the movies)
- vote_average (scale of 1 to 10)
- vote_count (the number of users who voted for the movie)
- **45467 rows**

2.7.3. TMDB Dataset

- **tmdb-movies.csv contains:** tmdb-movies.csv
  - id (movie Id)
  - imdb_id
  - Popularity (Popularity is a value that usually is updated/calculated daily when calculating it, the following things are taken into account: the site views, number of user ratings/watchlist/favourite additions, and release date.)
  - budget
  - revenue

- original_title
- cast
- homepage
- director
- tagline (Memorable phrase from the movie or that describes the movie)
- keywords
- overview (brief description of the movie)
- runtime
- genres
- production_companies
- release_date
- vote_count (Represents the number of people who gave their rating for the specific movie)
- vote_average (1 to 10)
- release_year
- budget_adj
- revenue_adj
- **10867 rows**

2.7.4. Yahoo Data Set

3 CSV files:

- yahoo_movie_rating.csv
- yahoo_movies.csv
- yahoo_users.csv

yahoo_movie_rating.csv contains:

- user_id
- movie_id
- rating (1 to 15)
- converted_rating (1 to 5)
- 10137 rows

yahoo_movies.csv contains:

- movie_id
- original_title
- Column3 (the movies are given a number in increasing order from 1 to 3952)
- 3884 rows

yahoo_users.csv contains:

- user_id
- user_birthyear
- gender (m or f)

2.8. Methods to measure the Accuracy

To make sure that a recommender system is optimal it needs to be accurate. Many authors including those referred to in the existing solution section have attempted to suggest ways of evaluating how good a system is. A system can be evaluated using the following approaches:.

- Given the datasets available that contain user demographic data we will train using the 1 million movielens data set and test on the yahoo data set.
- We will split the data from the movielens into into training and test data subsets. Train or run the application using a certain percentage of the users and test using a different subset. While there are programs to split data into training and test data here, we will arbitrarily we will take a subset of 10 users: assign 8 to the training set and 2 to the test set.
- Another option will be to try the same approach as above, but this time use the most up to date IMDB dataset which contains measurement of the popularity of a film over genre and age.

2.8.1. Confusion Matrix

To Test suitability the system will first be trained and then 100 users will be chosen at random from the test dataset. Movies will be recommended, and the "accuracy" of the measurement will be evaluated using the confusion matrix as illustrated in Figure 23. 1.

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Figure 23. 1 A generic confusion Matrix: showing the relation between True Positives, True Negatives, False Positives and False Negatives.

2.8.2. Mean Average Precision

Another method of measuring accuracy is the mean average precision. This is used with ranked ordered lists such as the recommendation of movies. It is normally applied if the choices further down the list are almost never chosen. The precision and recall for a recommender system can be determined using the following formulae illustrated in Figure 24. 1:

$$\text{recommender system precision:} \quad P = \frac{\#\text{ of our recommendations that are relevant}}{\#\text{ of items we recommended}}$$

$$\text{recommender system recall:} \quad r = \frac{\#\text{ of our recommendations that are relevant}}{\#\text{ of all the possible relevant items}}$$

Figure 24. 1 The precision and recall for recommender systems formulas.

The relevant items can be the actual films, although it is arguable better to use genres since the size is smaller, e.g. in the dataset users only made on average 20 recommendations out of about 3000 films so it could result in a type of cold start problem.

2.8.3. Personalisation

Another method to test the accuracy will be to personalise it to individual users. To test if a model recommends the same items to users. Ideally it should tailor the choice to individual users. So rather that looking for the degree of similarity between users or user demographic groups, the degree of similarity will be evaluated to see how personalised the system is. So, we will need to determine the cosine similarity. Since the recommender system used the cosine similarity a number of times it should be easy if there is sufficient time to adapt it to evaluate the personalisation of our system.

2.8.4. Intra-list Similarity

Using a list of movies that are recommended to each user or user demographic group: the genre(s) of each one is evaluated, and a similarity score based on the Genre Similarity is measured. If the system is recommending similar items to many single users or user groups, then the intra list similarity will be high. Again, since the item item genere similarity matrix is using a very similar concept it is hoped that it should be easy to adapt it for this purpose.

2.9. Alternative Existing Solutions to Your Problem

There are several movie recommendation systems out there. Many software developers have tried and are still trying to come up with the best movie recommendation system. However, no one was able to design the perfect movie recommendation system since none of the existing movie recommendation systems gives 100 % accuracy. Which is why coming up with a good recommendation system is still popular among software developers.

The following are some of the existent movie recommender systems:

1. [1] “Personalized Movie Recommendation (2009)” paper by Anan Liu, Yongdong Zhang, and Jintao Li proposed a hierarchical framework for personalized movie content. This framework

consists of “movie association and recommendation hierarchy” and “personalized movie navigation hierarchy”. For this project two experiments were conducted, one for each hierarchy used and 20 volunteers were used for each experiment. Both experiments gave favourable results. At the end of the project, they came to the conclusion that the hierarchical framework is very useful because it avoids problems such as the problems related to user preference collection, and it facilitates movie access with personalized recommendations.

This project will be useful for the project because it gives an idea of the possible algorithms and evaluation techniques that can be used. Also, it gives an insight into the issues which could be potentially faced throughout the project. Moreover, the result achieved from this project could be used to compare with the results which will be gained from the movie recommendation system.

2. [2] Phonexay Vilakone, Khamphaphone Xinchang, and Doo-Soon Park attempted to come up with efficient personalized movie recommendation system. They attempted to achieve that by combining different methods and comparing them with each other. For this project the following methods were used: k-clique, collaborative filtering, and collaborative filtering using the k-nearest neighbor. Their main goal for this project was to come up with a method which will give a better result than the other methods. After comparing the results achieved from their method against the results achieved from the other methods, it was found out that their methods gave a better result. In [2] Figure 19. 1fig 19 you can see the structure of the movie recommender system proposed in this project.

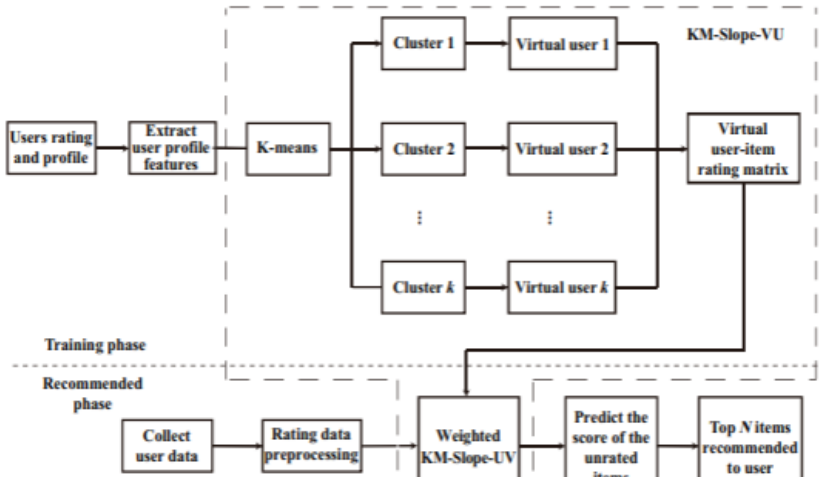
   This project concludes that the most optimal way to make a recommendation system more accurate is by combining the data mining method with the k-clique method. This paper is also very useful for the project because it gives some insight into how to combine different methods. Also, it introduced a few new algorithms which could be potentially used for the new movie recommender system.

[2] Figure 19. 1 Workflow of the proposed method

3. [3] Jiang Zhang, Yufeng Wang, Zhiyuan Yuan, and Qun Jin for the project tried to come up with a movie recommendation system which will solve two important issues: scalability and practical usage feedback and verification based on real implementation. For this project MovieWatch was created. MovieWatch is a web-based movie recommendation system, in [3] Figure 20. 1. You can see the framework of MovieWatch.

   MovieWatch used KM Slope VU collaborative filtering approach. At the end of their project, they were able to develop a collaborative filtering approach (KM Slope VU) which was able to improve one of the issues which is the scalability of movie recommendation system. It was not able to improve the second issue because some of the movies from the dataset were too old and the data set did not contain new movies. Although the result show high accuracy, it does not give the real result because of the above issues.

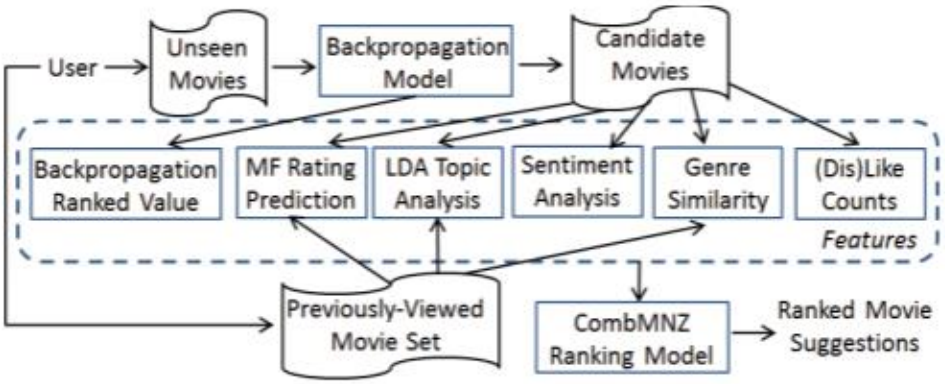
   This paper will be useful for my project because of its unique approaches to solve problems and the algorithms used. Also, a lot can be learned from their issues. Furthermore, the results they achieved can also be used. Moreover, some of the findings made during the project will be very useful when designing the new recommendation system such as being aware of time issues, scalability and irrelevant data.



[3] Figure 20. 1 The framework of MovieWatch

4. [4] You-Kai Ng goal was to create a personalized movie recommender system for children. The system is meant to filter out all the inappropriate content for children. You Kai Ng created MovReC, which used the following techniques: backpropagation classification, rating prediction, topic analysis, sentiment analysis and opinion mining, similarities in genres, like and dislike counts, and CombMNZ (which is a data fusion method for combining multiple ranked lists). After comparing MovReC with other movie recommendation system, MovReC's performance is higher. Also, it significantly outperforms other movie recommendation systems when it comes to prediction accuracy.

   This is a very impressive project because the results showed that this recommender system is more accurate than other existing recommending systems. This type of paper is very helpful for me because it uses new approaches into developing a recommender system. Also, compared to other movie recommender systems, MovReC combines a lot of techniques together. Furthermore, a lot can be learned from the structure of MovReC.



[4] Figure 21. 1 The children movie recommendation process of MovRec

2.10. Existing Final Year Projects

Project 1
Title: Football Data Mining, Result Prediction and Visualization

Student: Yahia Ragab

Description (brief):
An application that will help the users to make decisions on their Football bets in real time. This app will be displaying live predictions through web and mobile interface. The main goal of this project is to demonstrate how data analytics and data mining can be used for sports betting.

What is complex in this project:
Coming up with the best way to analyze the data. Also, the datasets should not be missing any piece of information. Since this app is going to be real time you need to make sure that the app wont crash in the middle of a match.

Technical architectures used:
Django, AWS, Node Package Manager, HTML, CSS, and Javascript

Explain the strengths and weaknesses of this project:
The strength of this project are that the project is very well structured. Many experiments were conducted which is very important. Also, the fact that there was a mobile app and a web app created.

Project 2

Title: Boppable – A Hybrid Mobile Application for Group Music Selection and Voting

Student: Emmet Doyle

Description (brief):
Booppable is a mobile application which will be build so that when you are going to a party you got to have a say as to what music will be played. This app will give an opportunity to both the host and the guest to choose what kind of music will be played, by voting or requesting a song. This app is a cross platform mobile application.

What is complex in this project:
To create a client application and a server application which will communicate with each other.

Technical architectures used:
PHP, MySQL, Django, AWS, Docker

Explain the strengths and weaknesses of this project:
The strength of this project is that it is made for both Android and iOS platforms. Another strength for this project is that different types of Testing were conducted.

3. System Design

3.1. Introduction

Following on from the existing software and research papers that we reviewed in the previous chapter, in this chapter the design of the system will be discussed. First the methodology used in this project will be outlined. After this I will give an overview of the system, along with Use Case diagrams to describe the functionality it offers. After this I will explain the Front-End, the Middle Tier, and the Back-End of my system.

3.2. Software Methodology

For this project I am planning to use the agile methodology. Throughout developing the movie recommendation system, I will be testing the result. If the results won't be accurate, I will have to make changes and then test it again. I will be repeating this process until I will be satisfied with the accuracy of the result. In other words, the testing process is very important for the development of the movie recommender systems. After researching several agile techniques, I found many useful approaches such as Cross-Industry Process for Data Mining (CRISP-DM), Analytics Solutions Unified Method for Data Mining (ASUM-DM), Feature driven development (FDD), Test driven development (TDD), and Scrum. All the listed above methodologies could be potentially used for this project. Since the movie recommender system which I intend to develop is dependent on constant testing I decided to use the TDD Agile methodology. I believe that this methodology is very practical and useful for my project.

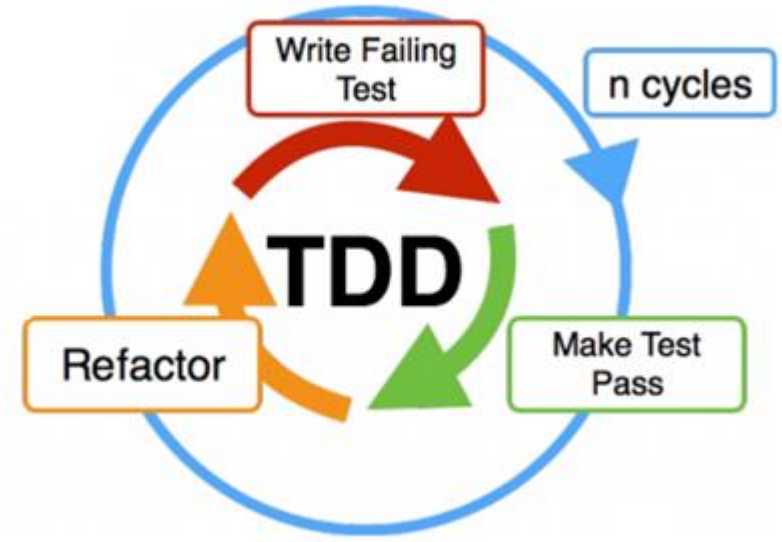


Figure 5. 1 The workflow of Testing Driven Development

3.3. Overview of System

The movie recommender system is going to be working in the following way. First a specific data set will be used. The data consists of a movie and features of the movie such as genre, language, duration, and rating. Then the data will be processed using two methods content-based filtering and a content-based filtering. Then the results achieved from each method will be combined using a combiner, which will turn it into hybrid filtering. Then a recommendation will be made.

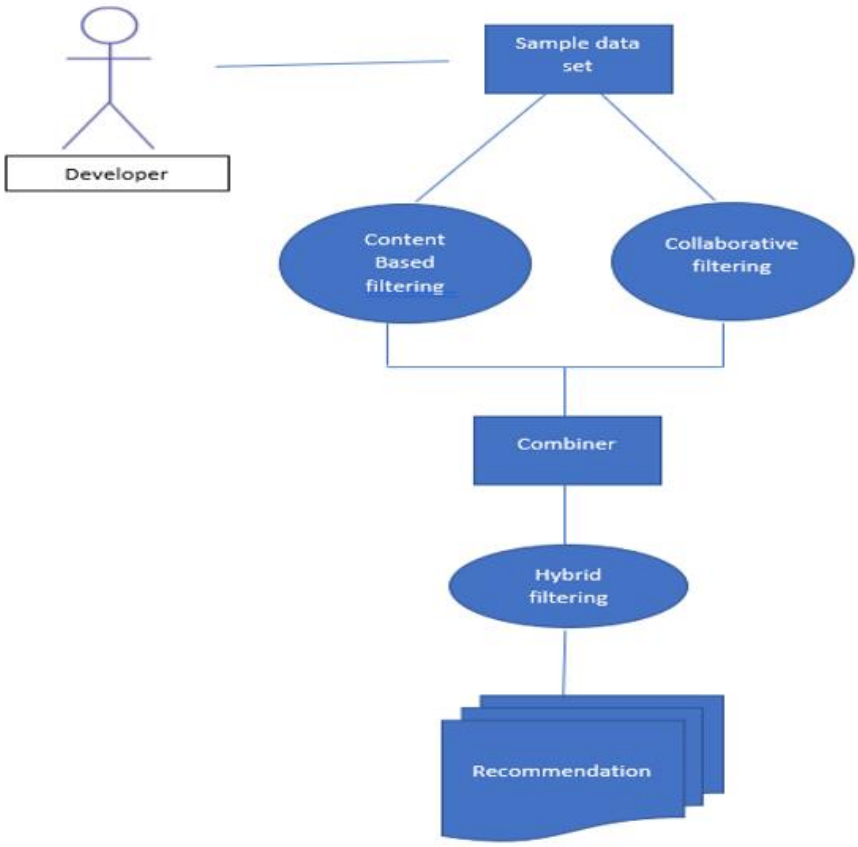


Figure 6. 1 A sample diagram of the Movie Recommender System

3.4. Front-End

Bellow you can see a simple diagram for the front-end of the system.

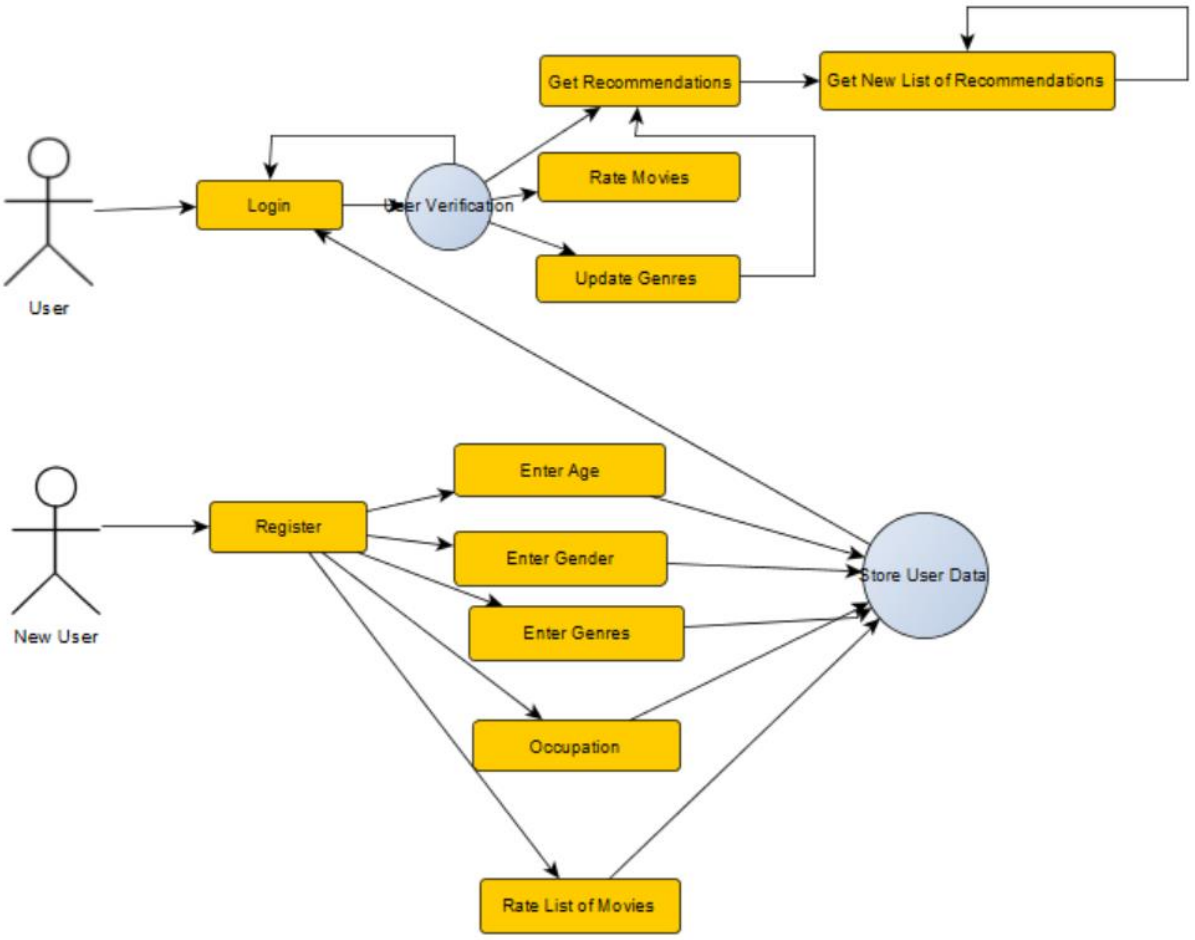


Figure 7 1 A sample diagram for the front-end

In Figure 7 1 there is a simple representation of how the front end will work. As we can see from the diagram a new user will have to register. During the registration process they have to provide their age, gender, occupation, favourite genres, and they will have to rate a list of movies. Once they will provide the mentioned information, the data will be stored and now they are users of the system. Every user registered in the system will have to log in, then the system will verify if they are existing users, if they were not found as existing users, they will be asked to log in again. Once the user is verified, then they will have a few options from which they can choose:

- They can get recommendations
- They can rate movies
- Change the genres they previously choose.

If the user selected the option to get recommendation, then they will be given a set of movies. If the user doesn't like the movies recommended or if they have previously watched the movies but there is not record of that on the system, then they can get a new set of movie recommendations. They can repeat that process several times until there will be no more movies to recommend. When they login to the system, and select the get recommendation option, that will generate a long list of movies. However, the system will be recommended only a set of movies. Then when they ask for a

new list of recommendations then they will be recommended the next set of movies from the list that was generated previously. This will be done until the end of the list will be reached.

If they will choose the option to rate the movies, then they will have to enter the name of the movie which they want to rate and the rating for that movie on a scale from 1 to 5. Once the user will enter the movie name, the system will search the existing movie data set and if the movie was not found then an appropriate message will be displayed to the user. Otherwise, the system will retrieve the movie ID, and it will add the user id, movie id and rating to the rating dataset. Then the predicted user item matrix will be updated.

If the user chooses the option to change the genres of the movies which they would like to get recommendation for, then they will be able to do that. After that they will be able to select the option to get recommendation for movies based on the genres they selected. This will form part of the get new recommendation.

3.5. Middle-Tier

The middle tier is very significant for this project. In this step the following will be created:

- user * user matrix,
- item * item matrix,
- user * item matrix using the matrix factorization.

With the help of the mentioned matrices, I will be able to make movie recommendations which is one of the main goals of this project.

The user * user matrix will be created using the following information: user's age, occupation, and gender.

The item * item matrix we can use various information about the movie, such as the movie's genre, cast, director, original language, and rating. An item vector: python cosine similarity to produce am item item matrix. An example of an item vector we can see in [23] Figure 15. 1

User –item matrix: uses the sparse user item from the rating csv file; performs matrix factorisation and produce the predicted user item matrix.

Once the user matrix and the item matrix and user-item matrices are created, then we will proceed to combining them to produce different sets of recommended movies. Based on e.g. user similarity, item similarity.

Using collaborative filtering and content-based filtering we will generate three lists of movies that could be recommended to the user. The three lists will then be combined in one list. This will be done in the following way, we will compare the three lists against each other, and then whenever there is a matching movie that movie will be added to the list.

The user*user, item*item, and user item matrices will be generated before the user logins. These matrices will be updated every week. The recommendation made to the users will be taken from the matrices generated. For that we will be using filtering, for example the movies will be filtered based on the movie genres.

The users will not be able to see how the recommendations are made. They will only see the recommendations which will be displayed in the interface. In order to connect the movie recommender system with the interface, the Django framework will be used.

3.5.1. How The User*User Matrix will be created:

- Choose the appropriate dataset. For this part a dataset which contains user demographics such as gender, age, and occupation, will be used. This data set will be loaded into the program, and it will become a Dataframe. This Dataframe will be used in the program so whenever a change will be made, it will apply only to the Dataframe and not the original dataset.

- Then each user demographic will be extracted into a list of user features. Then this list of features will be added to the Dataframe as a column. Each user will have their own list of features. So instead of having separate pieces of information we will be dealing with one list of features. Eg.

User ID	Gender	Age	Occupation
1	M	25-34	scientist
…	…	…	…

becomes →

user_data
M 25-34 scientist
…

- Then the new column "user_data" will be converted into a matrix of token counts. This is done using CountVectoryzer function. (***CountVectorizer is a library which is provided by the scikit-learn library in python. Its main purpose is to transform a given text into a vector on the basis of the count/frequency of each word that occurs in the entire text.***)

- Then the similarity between users will be measured using the cosine_similarity function. (***The cosine_similarity() function, calculates the similarity as the normalized dot product of X and Y: K(X, Y)= <X,Y>/(||X||*||Y||) . This library is provided by the sklearn.metrics.pairwise library in python. This library is used for 2D arrays***.)

- The similarity between users will create an array where each row represents a user and each column represents a user, and the value will represent the degree of similarity between users.

- In the diagram bellow there is a simple representation of the process explained above:

	Feature1(Gender)	Feature2(Age)	Feature3(Occupation)
User 1	M	25-34	scientist
User 2	F	35-44	scientist
User 3	M	25-34	student
User 4	F	25-34	programmer
User 5	F	18-24	writer
…	…	…	…

↓

	M	F	Under -18	18 - 24	25 - 34	35 - 44	…	Programmer	Writer	Student	Scientist	…
User 1	1	0	0	0	1	0	…	0	0	0	1	…
User 2	0	1	0	0	0	1	…	0	0	0	1	…
User 3	1	0	0	0	1	0	…	0	0	1	0	…
User 4	0	1	0	0	1	0	…	1	0	0	0	…
User 5	0	1	0	1	0	0	…	0	1	0	0	
…	…	…	…	…	…	…	…	…	…	…	…	…

	User 1	User 2	User 3	User 4	User 5	…
User 1	1	0.3	0.8	0.3	0	…
User 2	0.3	1	0	0.3	0.3	…
User 3	0.8	0	1	0.3	0	…
User 4	0.3	0.3	0.3	1	0.3	…
User 5	0	0.3	0	0.3	1	…
…	…	…	…	…	…	…

From this matrix it can be observed that whenever a user is compared to itself it will have a similarity score of 1. User1 and User5 don't have anything in common since their similarity score is 0. User1 and User3 have a similarity score of 0.8, this is a high similarity score therefore these users are quite similar. User3 and User4 have a similarity score of 0.3 this similarity score is low, this means that they are not very similar.

3.5.2. How the Movie*Movie (Item*Item) Matrix will be created:

- The first thing to be done to choose the appropriate Movie dataset. Then load it into the program as a data frame.
- Choose which features will be used to compare the movies. One feature or multiple features can be used. The genre is one of the common features of a movie, and it is one of the significant features when a user decides which movie they like to watch.
- Then the chosen features will be extracted, and the features will be combined into a new column in the DataFrame.
- After that the new column of features will be transform into a vector of tokens using the CountVectorizer() function.
- Then the degree of similarity between the users will be computed using cosine_similarity() function. And the results will be displayed into a matrix, where each row represents a movie,

and each column represents a movie, and the value will represent their similarity score. Bellow, is a simple representation of the process explained above:

	Feature1	…..
Movie1	Drama, Triller, Action, Horror.	….
Movie2	Drama, Romance, Comedy.	….
Movie3	Comedy, Romance.	….
….	….	….

	Drama	Triller	Horror	Comedy	Romance	Action	….
Movie1	1	1	1	0	0	1	…
Movie2	1	0	0	1	1	0	…
Movie3	0	0	0	1	1	0	…
……	…	…	…	…	…	…	…

	Movie1	Movie2	Movie3	…
Movie1	1	0.2	0	…
Movie2	0.2	1	0.8	…
Movie3	0	0.8	1	…
…	…	…	…	…

In the matrix above whenever a movie is compared to itself it will get a score of 1. Movie1 and Movie2 have a low similarity score of 0.2, the only thing they have in common is that they have Drama as a genre. Movie2 and Movie3 have a similarity score of 0.8 this score is high because both have Comedy and Romance as a genre. Movie1 and Movie3 have a similarity score of 0 that means that they have nothing in common.

3.5.3. How User*Movie (User*Item) Matrix will be created:

- The first step is to choose the appropriate rating dataset. Then load the dataset into the program, as a Dataframe.

- Then a sparse matrix will be created. This will be done using the ratings which was given to the movies by the users. However, not every user has rated every movie, so the matrix is going to be sparse. The rows in the matrix will represent the users and the columns will represent the movies and the value will be the rating given by the user to the movies. If a user didn't rate a movie, then a value of 0 will be assigned. Bellow you can see a simple example of a sparse matrix:

	Movie1	Movie2	Movie3	Movie4	Movie5	…
User1	5	0	0	0	0	…
User2	0	0	3	0	0	…
User3	0	0	0	0	4	…
User4	3	0	0	2	0	…
User5	0	0	0	4	0	…
…	…	…	…	…	…	…

In the above diagram where there is a 0 that means that the user didn't rate that movie. Wherever there is a value that means that the user did rate that movie

- The next step is to predict the rating which the users could potentially give to the movies. There are various ways to do that. For this project matrix factorization (SVD (Singular Value Decomposition)) will be used. This is done in order to transform the sparse matrix into a non-sparse matrix.

- However, using SVD is not enough, therefore it was decided to use SVD with SGD (Stochastic Gradient Descent). (***SGD is an optimization algorithm which is used to find the model parameters that correspond to the best fit between predicted and actual outputs. So SVD is used as a collaborative filtering technique which is used to get low rank factors of rating matrix and to reduces the dimensions and the SGD is used for optimization of the error objective function.***) Once the SVD with SGD will be applied we will be able to generate a non-sparse matrix. A simple representation example of such matrix can be seen bellow:

	Movie1	Movie2	Movie3	Movie4	Movie5	…
User1	5	4	3	4	3	…
User2	2	2	3	3	4	…
User3	2	3	5	2	4	…
User4	3	4	3	2	3	…
User5	3	2	3	4	4	…
…	…	…	…	…	…	…

The sparse matrix which was generate before will turn into a non-sparse matrix. These ratings are predicted ratings. The values are potential ratings which the user would potentially rate the movies if they would've watched the movies.

- The generated non sparse user*movie(user*item) matrix is the matrix which will be used to recommend movies to users.

3.5.4. How The Recommendation List Will Be Generated:

3.5.4.1. List1:

- For this the matrices which have been generated before will be used. The user*user matrix and the movie*movie(item*item) matrix.
  Start with the user*user matrix. The matrix looks like this:

	User1	User2	User3	User4	User5	…
User1	1	0	0.5	1	0.3	…
User2	0	1	0.3	0	0.2	…

User3	0.5	0.3	1	0.2	0.5	…
User4	1	0	0.2	1	0	…
User5	0.3	0.2	0.5	0	1	…
….	…	…	…	…	…	…

Before going to the next part, a record of the index number of each user will be made.

Then the users who are similar to a selected user will be found. For example, all the users who are similar to User3 will be found. The first step is to go to the user*user matrix and find the row which has all the similarity score between user3 and the rest of the users in the dataset.

	User1	User2	User3	User4	User5	…
User1	1	0	0.5	1	0.3	…
User2	0	1	0.3	0	0.2	…
User3	0.5	0.3	1	0.2	0.5	…
User4	1	0	0.2	1	0	…
User5	0.3	0.2	0.5	0	1	…
….	…	…	…	…	…	…

Then this row will be extracted into an array/list.

User3	0.5	0.3	1	0.2	0.5	…

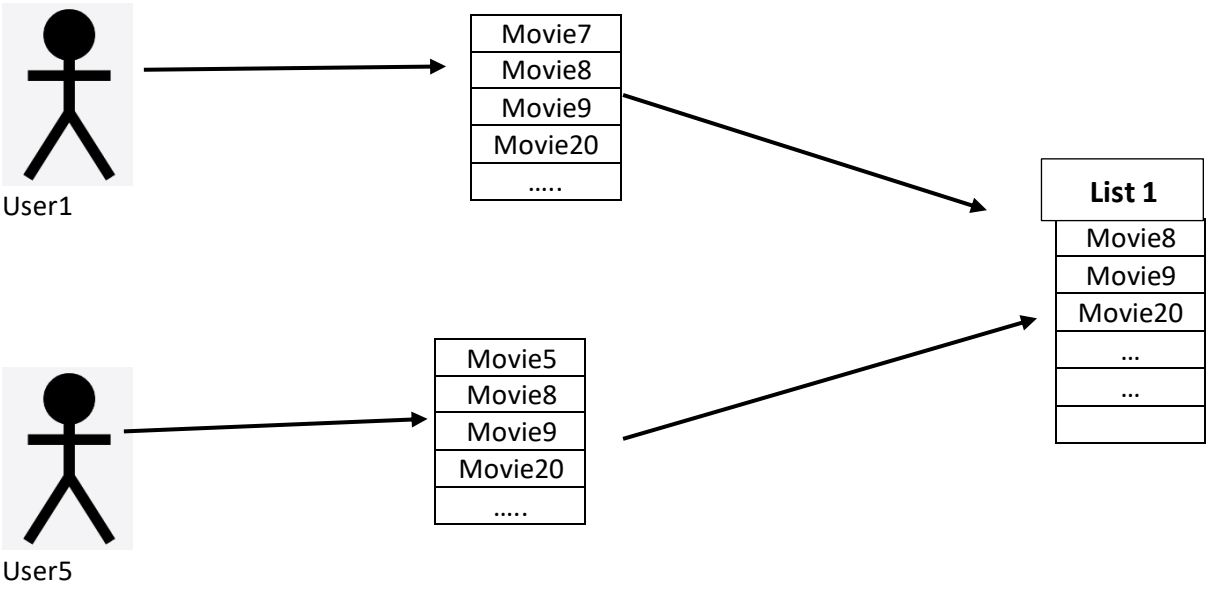
Next step will be to sort this array/list in decreasing order. The users with the highest similarity should be at the top.

User3	1	0.5	0.5	0.5	0.5	…

Then only the users with the highest similarity will be selected, their user ids and similarity scores will be saved into a list.

User Id	Similarity Score
User3	1
User1	0.5
User5	0.5
User10	0.5
User223	0.5
…	…

Then the users will be taken one by one, and then the movies which movies which they have seen previously will be looked at. Only the movies with a high rating will be considered for the next part.



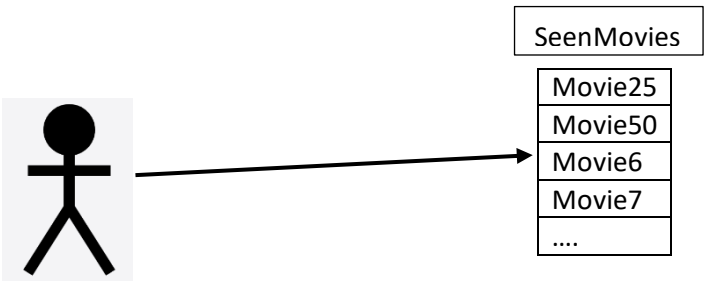
The movies which were watched by user3 will not be considered. Then the movies which were watched by all the similar users will be checked. If there is a movie that was watched by the majority of users, then this movie will be added to a list. In the above diagram we can see that both users have watched movie8, movie9, and movie20, assuming that there are other users in the list who have watched these movies, then these movies will be added to the list which will be used to recommend movies to the user3.

How do we know which user has watched which movie?

It will be assumed that if a user has rated a movie that means that the user has seen the movie. We will add a row to the dataframe which will state whether a user has seen a movie or not so wherever there is a rating to a movie that means that the user has seen the movie and it will get a value of 1 and if the user hasn't rated the movie, then it will get a value of 0.

3.5.4.2. List2:

- To generate the 2nd list the movies which were seen by user3 will be used. Lets say user3 has a list of movies which he has seen.



Then only the movies that had the highest ratings will be used. The movies will be taken one by one and using the movie*movie(item*item) matrix the most similar movies will be found. This will be done in the following way:

Lets say Movie25 mill be used. Movie25 will be found in the movie*movie(item*item) matrix and the row which contains the similarity score between movie25 and the rest of the movies.

	Movie1	Movie2	Movie3	…
……	….	…..	….	…
Movie24	0.2	1	0.8	…
Movie25	0	0.8	1	…
…	…	…	…	…

Then this row will be extracted into a list/array.

Movie25	0	0.8	1	……

Then row will be sorted in a decreasing order. Then movies with the highest similarity will be at the top of the list.

Movie25	1	0.8	0	……

Then the most similar movies will be selected and added to a list. The same thing will be done to the rest of the movies, and the most similar movies will be added to the second list. However, before adding the movies to the list, it will be first check if the movie exists in the list. If a movie is in the list already then that means that it won't be added to the list. This will be done in order to avoid having duplicated movies. This list will become the second list, which will used to make recommendations.

3.5.4.3. List3:

To generate this list the user*movie(item) matrix will be used.

- To generate this list the user*movie(item) matrix will be used.

Find the user to which the movies will be recommended. Find it in the user*movie matrix. For this example, movies will be recommended to user3.

	Movie1	Movie2	Movie3	Movie4	Movie5	…
User1	5	4	3	4	3	…
User2	2	2	3	3	4	…
User3	2	3	5	2	4	…
User4	3	4	3	2	3	…
User5	3	2	3	4	4	…
…	…	…	…	…	…	…

Then that row is extracted as a list/array. There will be a record of the index for each rating.

User3	2	3	5	2	4	…

Sort the list in decreasing order. The movies with the highest score should be at the top of the list.

User3	5	4	3	2	2	…

At this point there will be a big list. All the seen movies will be filtered. For this example, it is known that User3 has rated Movie5(Movie 5 has a rating of 4), therefore movie 5 will be filtered out.

User3	5	3	2	2	…

Once the seen movies are filtered. The list will be filtered again, because only the movies that are most likely to be liked by the user are wanted on the list, which will be recommended, all the movies that have a rating lower than three will be eliminated. So, in the list we will be having the movies which will have a rating of 3 or higher.

User3	5	3	…

Once all the movies which are not wanted are filtered, the rating of the movie and Movie Id will be linked back. When the list is extracted from the array, only the ratings of the movies and not the movies themselves are used. However, the index of each move is being tracked. Therefore, it is possible to link/map the movies index back to its movie id. After that each movie can be added to a list and this list will become the third list.

3.5.4.4. Combining the lists together:

Once the three lists are generated, all the unwanted movies are filtered and the wanted movies are moved to one list, which will be the list of movies to be recommended to the user.

Then it will be checked which movies are in all three lists. Those movies will be added to the final list. Then it will be checked which movies are in two of the three lists, those movies will also be added to the list. The order of the movies will be the following, the movies which are in all three lists will be at the top and the movies which are in two of the three lists will follow after. The order of the movies will not be based on their rating score because it was dealt with all the low ratings when generating the lists so there won't be movies that have a rating lower than 3. Therefore, the order of the movies is going to follow the order in which each movie was added to the list.

3.6. Back-End

The back end is another major part of this project, it consists of the datasets which will be used. All the filles used for this project will be kept in one folder because having all the datasets in one folder is more convenient than having each dataset in an individual folder. The folder will contain all the

filles from the IMDB dataset, TMDB dataset, Movielens dataset, and Yahoo dataset. Apart from the datasets which will be used for the recommender system, it is possible that each matrix will be saved in a separate file.

There will be the following databases:

- **User-Database**: This will be a .csv file containing the user information such as user id, age, genre occupation.
- **Movie-Database**: This will be a .csv file containing the movie information such as genre, actors, director, etc.
- **Rating-Database**: A .csv file containing the ratings of all the movies. This file will contain, movie Id, user Id and rating of the movie.

For this project we will need to add information to the datasets. That will be done in the following way. Whenever a new user will register, the information which they will provide will be stored in our csv files. The information such as user age, gender, and occupation will be stored at the end of the user .csv file. This is done in the form of adding a row to the file.

The user will also provide ratings for movies. That information will be stored in the rating .csv file. The existing user will also enter rating form movies, their data will also be stored in the rating .csv files. The existing users will be able to provide rating only for the movies that exist in the database.

3.7. Conclusion

In this chapter a sample the design of the movie recommender system was presented. First, the TDD methodology was discussed as the approach to be used in this project. Followed by a description of the movie recommendation system and sample design diagram of the system. The Front-End, Middle -Tier and Back -End were also discussed. The system which will be developed, will be focused more on the Middle - Tier and Back – End Tier, than on the Front – End.

4. Prototype Development

4.1. Introduction

In this chapter the development of the prototype will be discussed. I will give an overview of the prototype development

4.2. Prototype Development

4.2.2. Implemented System Architecture

In Figure 1 there is a representation of how lists one was generated. In order to generate this list, the user * user matrix Figure 5 was used. This matrix contains the measurements of the similarity between all the users that existed in the movielens dataset. To generate this matrix user demographics such as gender, age, occupation was used. By using the demographics instead of just using the ratings, the sparsity was reduced.

Then a user id was picked, out of the user *user matrix the user id's column is extracted, this column gives a list of all the similar users. This list is filtered and sorted to get the most similar users. Then for each similar user, it will be found which movies they have seen. This will be achieved using the user*item(movie) matrix. Then lists of movies seen by the most similar users to the chosen users will be generated. After that the lists will be filtered by checking if there are movies in all the lists. The movies which are common to all he lists or to the majority of the lists will created List1.

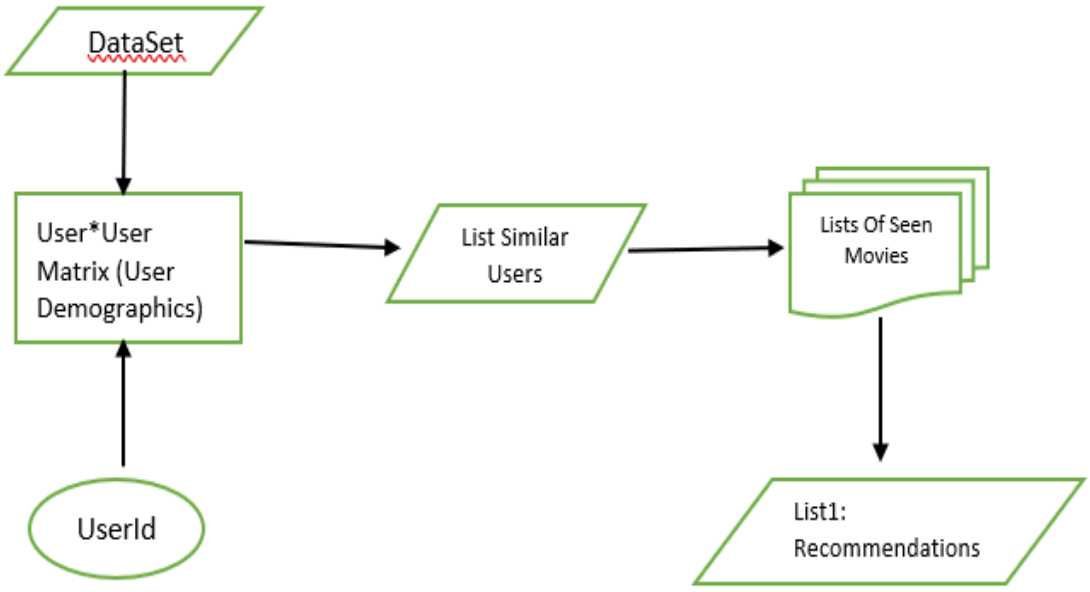


Figure 2, shows how Lists3 is generated. Just like list 1, list 3 is also generated using collaborative filtering. This approach is based on the concept matrix factorization. Although there is known information about the users and the matrix is generated. The matrix is very sparse (eg Figure 11) since user will rate a small number of movies, for example on an average a user might rate 20 movies out of 6000. To overcome this problem, a type of machine learning technique which is similar to neural networks was implemented. This technique is called matrix factorization.

Matrix factorization is used to predict ratings for movies. Overtime with what is called an epoch, the learning rate error starts to decrease, or the system learns to learn better. Therefore, over a large number of learning steps called epoch, it is possible to come up with a user matrix which is filled with predicted ratings (e.g. Figure 12). The predicted ratings are ratings for movies which were not seen by users previously but are given ratings based on users' previously rated movies. Once this matrix is generated a user ID is picked, then go to the corresponding row within the matrix, a list of movies is extracted. The movies with the highest rating or the highest predicted rating are picked.

Figure 1 Implemented System Architecture: Collaborative Filtering. How list 1 is generated.

They are added to List3.

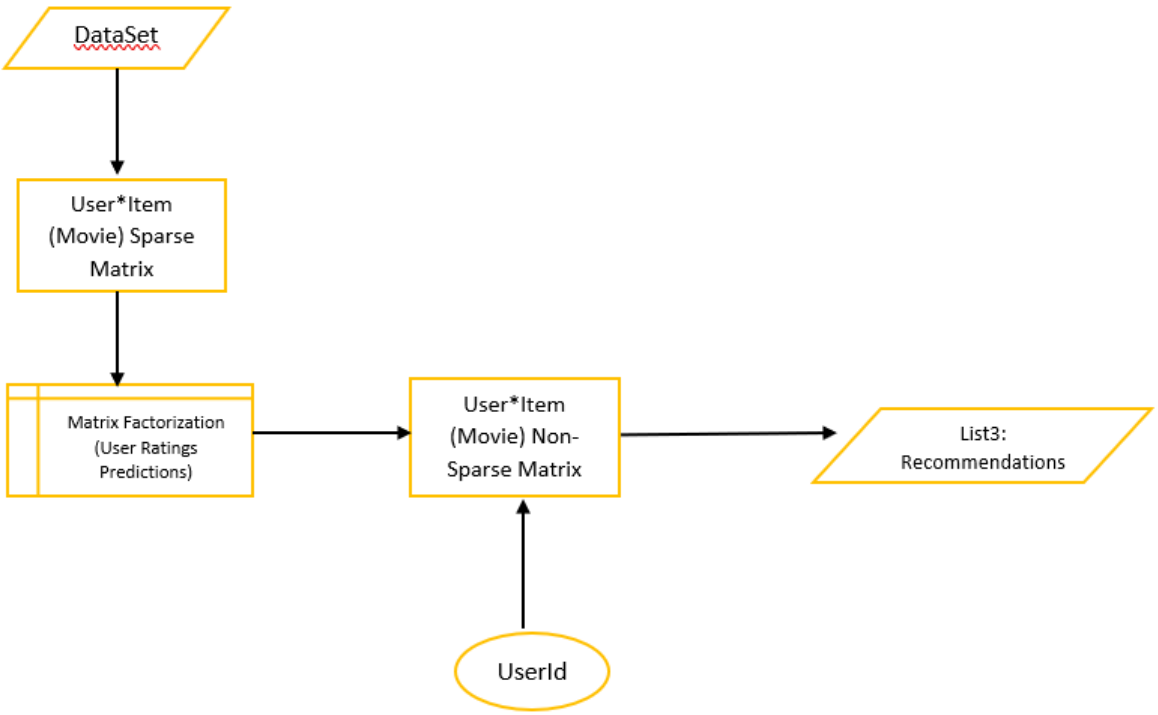


Figure 2 Implemented System Architecture: Collaborative Filtering. How list 3 is generated.

Figure 3 shows how List2 is generated. This list is generated using content-based filtering which is a different approach to collaborative filtering. In this case, just like before, it is looked at the list of movies using the similarity matrix generated for this example. This matrix Figure 6 is generated using genres only. Then for the user picked, go to the specific row, a list of movies seen by the user is extracted. For each movie, go to the movie*movie matrix, and for each movie, extract movies that were similar to the movie seen by the user. This eventually turn into lists of similar movies that are seen by the user. Then the movies that are common across all the lists will are extracted, such as the movies that are similar to all the movies previously seen by the users. This is how list 2 was generated.

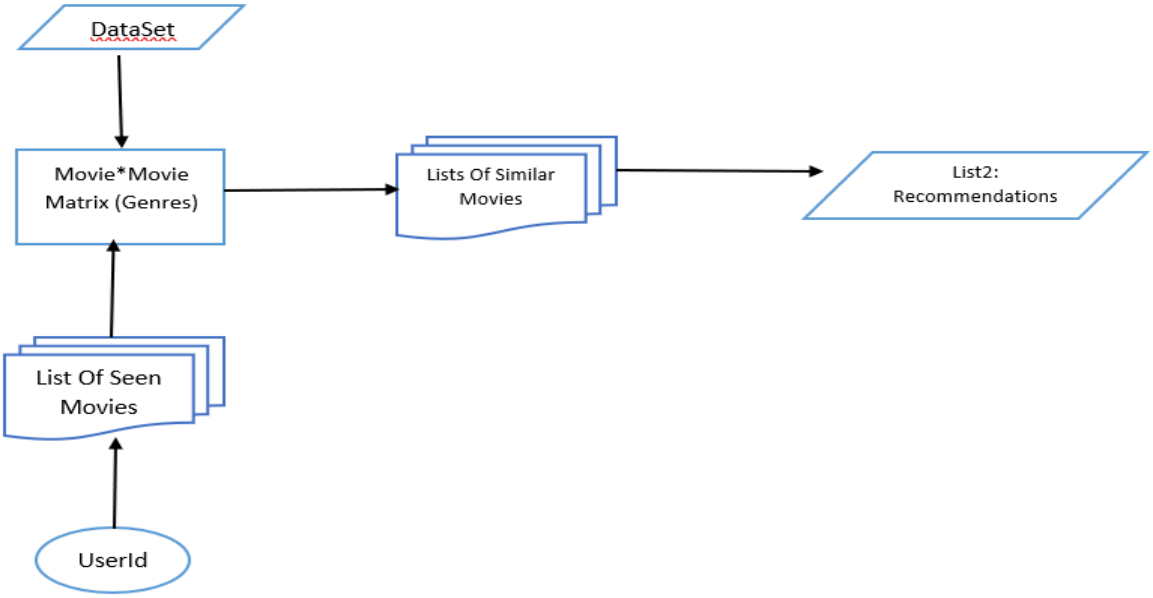


Figure 3 Implemented System Architecture: Content Based Filtering. How list 2 is generated.

Once the 3 lists are generated, the lists are combined together by finding which movies are common to all. This becomes the final recommendation list which will be recommended to the user. We can see an example of this in Figure 7, this list was generate for user with an userId 2, this list was generated using List 1 in Figure 8, List 2 Figure 9 and List 3 Figure 10.

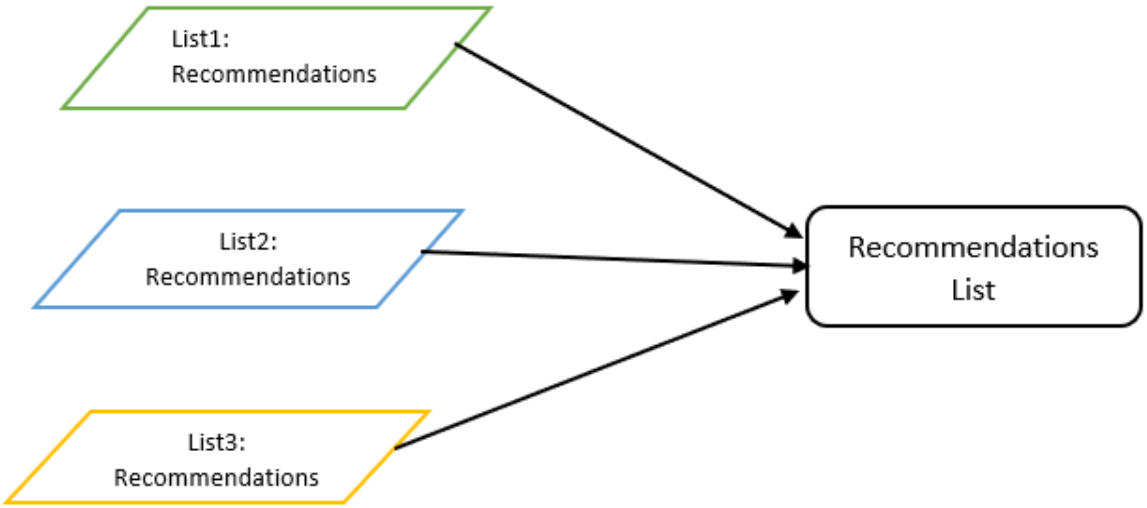


Figure 4 Implemented System Architecture: How the lists are being joined together.

```
User*User Matrix:
[[1.         0.         0.         ... 0.         0.         0.        ]
 [0.         1.         0.         ... 0.40824829 0.         0.        ]
 [0.         0.         1.         ... 0.         0.         0.57735027]
 ...
 [0.         0.40824829 0.         ... 1.         0.         0.        ]
 [0.         0.         0.         ... 0.         1.         0.        ]
 [0.         0.         0.57735027 ... 0.         0.         1.        ]]
```

*Figure 5 Generated User*User Matrix*

```
Movie*Movie Matrix:
[[1.         0.         0.         ... 1.         1.         0.70710678]
 [0.         1.         0.         ... 0.         0.         0.        ]
 [0.         0.         1.         ... 0.         0.         0.        ]
 ...
 [1.         0.         0.         ... 1.         1.         0.70710678]
 [1.         0.         0.         ... 1.         1.         0.70710678]
 [0.70710678 0.         0.         ... 0.70710678 0.70710678 1.        ]]
```

*Figure 6 Generated Movie*Movie Matrix*

Figure 7 Generated Solution: The final recommendation list for user 2

```
User ID:  2

Movie Recomendations:
Ed Wood (1994)
Stand by Me (1986)
Shawshank Redemption, The (1994)
Saving Private Ryan (1998)
Shakespeare in Love (1998)
Bridge on the River Kwai, The (1957)
American Beauty (1999)
Gone with the Wind (1939)
Being John Malkovich (1999)
Green Mile, The (1999)
Strictly Ballroom (1992)
Godfather, The (1972)
Insider, The (1999)
Babe (1995)
Amadeus (1984)
Rushmore (1998)
Jurassic Park (1993)
Raiders of the Lost Ark (1981)
Godfather: Part II, The (1974)
Braveheart (1995)
Dances with Wolves (1990)
Moonstruck (1987)
Magnolia (1999)
Silence of the Lambs, The (1991)
Manhattan Murder Mystery (1993)
Waiting for Guffman (1996)
Forrest Gump (1994)
Titanic (1997)
Gladiator (2000)
Good Will Hunting (1997)
Schindler's List (1993)
Election (1999)
As Good As It Gets (1997)
Lost World: Jurassic Park, The (1997)
Grosse Pointe Blank (1997)
L.A. Confidential (1997)
All About My Mother (Todo Sobre Mi Madre) (1999)
```

Figure 8 Generated Solution: List 1 for user 2 for the data set

```
Movie Recomendations:
Dangerous Minds (1995)
Promise, The (La Promesse) (1996)
Two Friends (1986)
Costa Brava (1946)
Restoration (1995)
Anna (1996)
Nixon (1995)
Othello (1995)
Now and Then (1995)
Sadness of Sex, The (1995)
Hoogste tijd (1995)
Dead Man Walking (1995)
Cry, the Beloved Country (1995)
Venice/Venice (1992)
Low Life, The (1994)
Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
Clubland (1998)
Lamerica (1994)
Get Over It (1996)
Invitation, The (Zaproszenie) (1986)
```

Figure 9 Generated Solution: List 2 for userId 2 from the dataset

```
Star Wars: Episode IV - A New Hope (1977)
Star Wars: Episode V - The Empire Strikes Back (1980)
Star Wars: Episode VI - Return of the Jedi (1983)
Braveheart (1995)
Jurassic Park (1993)
Raiders of the Lost Ark (1981)
Fugitive, The (1993)
Sanjuro (1962)
Forrest Gump (1994)
Sixth Sense, The (1999)
Schindler's List (1993)
Matrix, The (1999)
Shawshank Redemption, The (1994)
Cold Fever (Á köldum klaka) (1994)
Indiana Jones and the Last Crusade (1989)
Back to the Future (1985)
Gladiator (2000)
Saving Private Ryan (1998)
Terminator, The (1984)
Grand Illusion (Grande illusion, La) (1937)
```

Figure 10 Generated Solution: List 2 for userid 2

Movie_id	1	2	3	4	5	...	3948	3949	3950	3951	3952
User_id						...					
1	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
3228	0.0	4.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3229	3.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	4.0
3230	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3231	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3232	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

Figure 11 Generated Solution: Sparse User Item Matrix

The time to complete the Matrix factorisation is: 11728.064273357391

	0	1	2	...	3617	3618	3619
0	4.234119	3.216595	4.548122	...	3.674749	0.962031	0.969116
1	3.879027	2.978683	3.965277	...	3.228428	0.840061	0.838578
2	4.204781	3.306580	3.806531	...	3.162294	0.809804	0.788727
3	3.666509	2.793723	3.885716	...	3.145982	0.822261	0.826310
4	5.067310	3.966587	4.702937	...	3.890238	0.999611	0.978786
5	4.109258	3.245204	3.633140	...	3.030844	0.773590	0.749551
6	4.125078	3.186008	4.100534	...	3.353473	0.869514	0.863333
7	4.595081	3.608779	4.189756	...	3.476327	0.891100	0.869250
8	4.110845	3.260714	3.544379	...	2.970177	0.755406	0.727788
9	4.123219	3.179833	4.128657	...	3.372517	0.875266	0.870274

Figure 12 Generated Solution: Non-sparse User-Item Matrix

4.2.3. Web Application

The focus of this project is not the front end. The intended approach was to develop a web app which will connect the recommender system described above, to the web application and then the users will be able to register, login, get recommendations, and rate movies. However due to the complexity of connecting all the python scripts and all the required libraries and dependencies, as well as the time constraints it was not implemented.

The only implemented and working parts are the login, and registration page. So the user can register and login to the system.

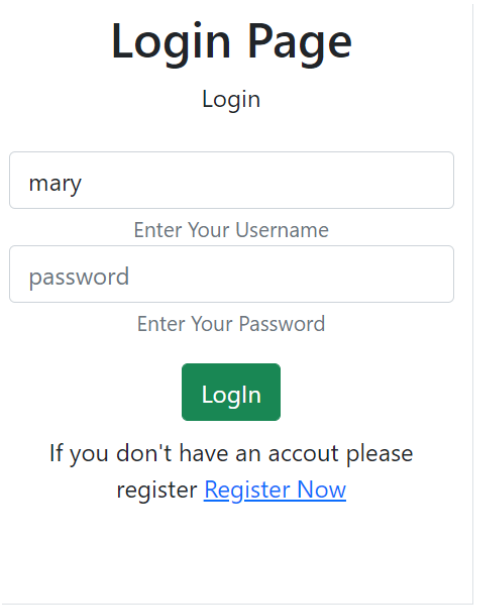


Figure 13 Generated Login Page

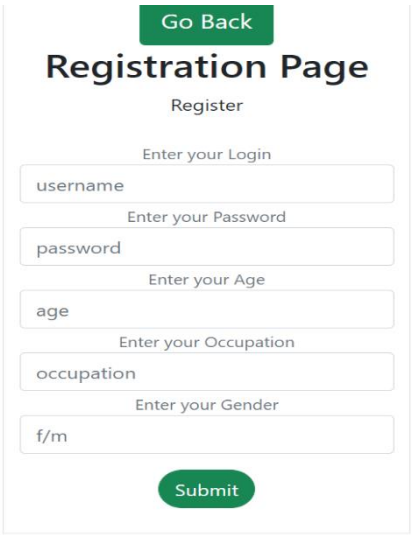


Figure 14 Generated Registration Page

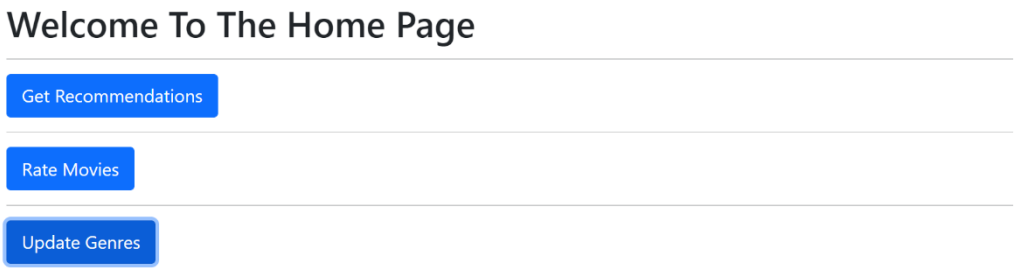


Figure 15 Generated Home Page

4.6. Conclusions

In this chapter the prototype development was explained. All the steps taken to generate the recommender system, as well as how the collaborative, and content-based filtering were implemented. It is also mentioned which parts were implemented successfully and which parts of the system where not implemented successfully.

5. Testing

5.1. Introduction

This chapter presents the development and testing process that was undertaken as part of this project. The type of testing that was done and the time of testing. Also, the challenges which were encountered during the testing phase. As well as the type of testing that was intended to implement but unfortunately it was not possible to implements because of the time constraints.

5.2. Testing

The testing was done on the Matrix Factorization, which is part of the collaborative filtering, the user *user matrix, which is part of the collaborative filtering, and on the movie*movie matrix which is part of the content-based filtering for this project.

5.2.1 Matrix Factorization

Testing for matrix factorization involved challenges such as time constraints and calculation complexity. To predict the ratings of a movies, the matrix factorization must learn and come with a prediction. Due to the large data sets it takes time for the system with large data sets to learn a prediction rating for all the users and for all the movies. Therefore, to learn it took time.

Another problem which was encountered with matrix factorization was the accuracy of the predictions, the more epoch that were carried out the better the accuracy of the prediction. For this we reduced the learning rate therefore improved the accuracy. Then it was needed to find balance between the time it was needed to run the programme and to ensure that it was as accurate as possible, or it is able to learn as sufficiently as possible.

For testing both large datasets and small data sets were used. The system was tested using 5 epoch, 10 epochs, 20 epochs, 50epochs, 100epochs, and 1000 epochs. While testing it was found that for very large amounts of epochs it was giving null values or a runtime overflow error. This occurred since every time the number of epochs increases the calculations become more complex. This error was given because of the complexity of the calculations.

This error in Python indicates that an arithmetic operation has exceeded the limits of the Python runtime. This is due to the use of large float values. The first time this error was encountered it was after running 30 epochs. An attempt to solve this problem was to use the expit function. *Expit is part of the scipy.special.expit library. It is also known as the logistic sigmoid function. What it does it basically it takes a ndarray as an input, applies the expit to element-wise. Then it outputs another ndarray of the same shape as the array which was taken as the input.* Once this function was applied the warning has disappeared. However, it appeared again as the number of epochs were increased. Then the warning appearing when the system was tested for around 127 epochs. Due to this being a warning the matrix factorization didn't stop its calculations, instead it kept going. After analysing the results achieved it seems that every time this warning was encountered, it will end up giving null predictions. Unfortunately, it was not possible to find the solution to this problem.

Then it was found that for the low epoch values there was not enough similarities, basically similar similarities and it was not learning enough. The optimal number of epochs to use seemed to be around 100 epochs. This led to one of our biggest challenges, which is execution time.

To run matrix factorization using 100 epochs it was taking around 12 hours, on a laptop with 8 GB of RAM. It was found that as we start to use a different laptop with a larger number of RAM the execution time has decreased. On a laptop with 16 GB of RAM it took around 6 hours. This again was going to be a problem as to how frequently run the Matrix Factorization will be executed. As users are running the system it should be taken into consideration if the matrix should be run every day every week or maybe every month.

Bellow are a few examples of results gained while testing the Matrix Factorization on the large dataset and on the smaller data set using different number of epochs. The results in each sample result displays a list of movie ids which are to be recommended to users.:

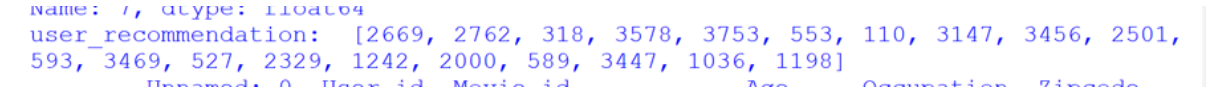


Figure 16 MF Large Dataset Using 100 epochs

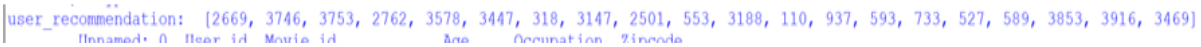


Figure 17 MF Large Dataset Using 50 epochs

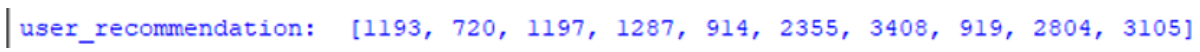


Figure 18 MF Large Dataset Using 10 epochs

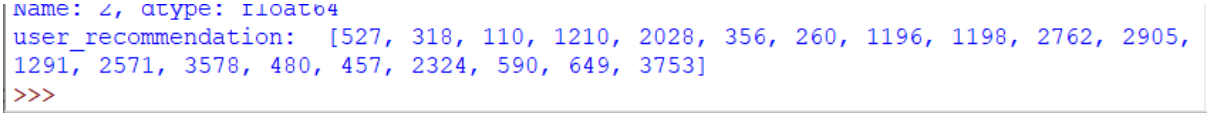


Figure 19 MF Small Dataset Using 10 epochs

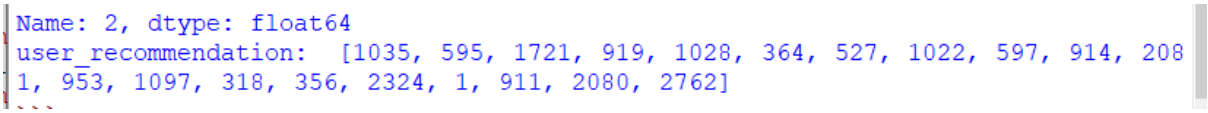


Figure 20 MF Small Dataset Using 50 epochs

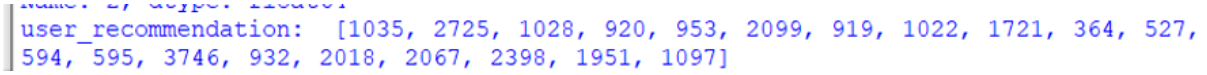


Figure 21 MF Small Dataset Using 75 epochs

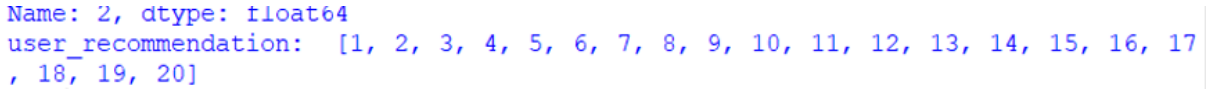


Figure 22 MF Result gained when there were RuntimeWarnings. For both the small dataset and the large dataset, every time there was a runtime error the movies recommended were the first 20 movies.

5.2.2. User Matrix and Movie Matrix

Testing was also done on User*User Matrix and Movie * Movie Matrix. To get List1 the user*user matrix was used, and to get List2 the movie*movie matrix was used. These two lists do not use any

prediction, so the results were expected to be identical every time each program was run and each list generated. So, what was tested in this particular case was if there are a number of different users, but they are identical across three demographic features or attributes. The results achieved must be similarity in terms of what was recommended. The same should be done for two attributes in common. Then it is meant to be tested for one attribute in common and finally test if there are no attributes in common. Once that was done the testing was meant to check for intra-list similarity but unfortunately there was not enough time to do that, but this was the intended approach.

Another part of testing was to look at each of the 3 lists, list 1, list 2, and list 3. Once again check for some degree of intra-list similarity. Unfortunately, this part was also not implemented.

5.3. Evaluation

In terms of evaluation of the system, it was intended to ideally evaluate the completed system. For the evaluation, a group of users would be asked to test the system functionality. They will check if they can register and log in, rate movies, get recommendations, etc. Then the list of movies that will be recommended will be evaluated. Or the users will have to evaluate a list of movies that were recommended. They would have to say whether they were happy or not with the recommendations. This is the kind of evaluation and personalisation intended for the system. However, because of the fact that the system was not fully developed, it was impossible to go through with any type of evaluation.

5.4. Conclusion

In this chapter it was discussed the testing which was done. The testing which was meant to be done. The challenges which were encountered as well as some of the limitations. As mentioned earlier not all the intended testing was done, since some of the testing required other part to be done first, and some testing was not finished because of time constraints.

6. Future Work

6.1. Introduction

This chapter presents the future work. What parts should be improved, and which parts should be implemented in the future.

6.2. Plans and Future Work

- The first thing for the future work will be to try and work on the algorithms and the limitations of the calculation, in order to improve them.
- The second thing is to attempt using larger part of the dataset/datasets, by using more fields when generating the matrices. As well as try to use more up to date data.
- The third plan for the future work is to find ways to reduce the cold start problem. For example, the users might be asked to rate previously viewed movies, add registration time for each user. Also, the user might be asked to enter their favourite actor, actress, director, etc. This will provide more information about the users and hopefully reduce the cold start problem for users.
- Attempt to reduce the cold start problem for movies.
- Implement the recommendation interface. First implement the interface as a standalone application, fully tested it. Then develop it into a web application.
- Use the stand-alone application to test the system. For this type of testing several users from different backgrounds and age groups will be asked to test and evaluate the system.
- Then improve, the overall system.

6.3 Conclusion

In this chapter all the plans for future work were discussed. Which includes all the work which need to be done further. As well as how the system can be improved.

Bibliography

1. Anan Liu, Yongdong Zhang, Jintao Li. (2009) Personalised Movie Recommendation [Internet] [cited 2020 Oct 10]. Available from: http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/Web/People/liuanan/MM09.pdf

2. Phonexay Vilkone, Khamphaphone Xinchang, Doo-Soon Park. (2019). Personalized Movie Recommendation System Combining Data Mining with the k-Clique Methods [Internet][cited 2020 Oct 10] Available from: http://jips-k.org/full-text/299

3. Jiang Yufeng Wang, Zhiyuan Yuan, Qun Jin (2019). Personalized Real – Time Movie Recommendation System: Practical Prototype and Evaluation [Internet] [cited 2020 Oct 12] Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8821512

4. You – Kai Ng. (2017). MovRec: A personalized Movie Recommendation System for Children Based on Online Movie Features [Internet][cited 2020 Oct 12] Available from: https://faculty.cs.byu.edu/~ng/papers/IJWIS.pdf

5. IBM Cloud Education (2020). Machine Learning [Internet][cited 2020 Nov 14] Available from: https://www.ibm.com/cloud/learn/machine-learning

6. Kashish Chugh. (2019). Types of Machine Learning Techniques. [Internet][cited 2020 Nov 14] Available from: https://medium.com/@chughkashish12/types-of-machine-learning-techniques-febd66fc87b9

7. MathWorks, What Is Machine Learning? 3 things you need to know. [Internet][cited 2020 Oct 29] Available from: https://uk.mathworks.com/discovery/machine-learning.html

8. Semih Yagcioglu, (May 18, 2020) Classical Examples of Supervised vs. Unsupervised Learning in Machine Learning [Internet][cited 2020 Oct 29] Available from: https://www.springboard.com/blog/ip-machine-learning-unsupervised-learning-supervised-learning/#:~:text=Another%20great%20example%20of%20supervised,tweet%20or%20a%20product%20review

9. Jason Bronwlee, (March 16) Supervised and Unsupervised Machine Learning Algorithms [Internet] [cited 2020 Oct 30] Available from: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

10. Unsupervised Machine Learning: What is, Algorithms, Example,[Internet][cited 2020 Oct 30] Available from: https://www.guru99.com/unsupervised-machine-learning.html

11. Alibaba Cloud, (Aug 7, 2019), A Closer Look into Major Types of Machine Learning Models [Internet][cited 2020 Nov 1] Available from: https://becominghuman.ai/a-closer-look-into-the-major-types-of-machine-learning-models-77164a47012

12. Blanzej Osinki and Konrad Budek, (July 5, 2018) What is reinforced learning? The complete guide [Internet][cited 2020 Nov 4] Available from: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

13. Badreesh Shetty, (July 24, 2019) AN IN-DEPTH GUIDE TO HOW RECOMMENDER SYSTEMS WORK [Internet][cited 2020 Nov 6] Available from: https://builtin.com/data-science/recommender-systems

14. Gaston Rodriguez, (2019) Introduction to Recommender Systems in 2019 [Internet][cited 2020 Oct 25] Available from: https://tryolabs.com/blog/introduction-to-recommender-systems/

15. Aditya Sharma, (May 29th, 2020) Beginner Tutorial: Recommender Systems in Python [Internet][cited 2020 Oct 25] Available from: https://www.datacamp.com/community/tutorials/recommender-systems-python

16. Crossing Minds, What are today's top recommendation engine algorithms? [Internet][cited 2020 Oct 25] Available from: https://itnext.io/what-are-the-top-recommendation-engine-algorithms-used-nowadays-646f588ce639

17. Baptiste Rocca, (Jun 3, 2019) Introduction to recommender systems [Internet][cited 2020 Nov 15] Available from: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

18. Ezra Soterion Nugroho, (1/10/2020) Movie Recommendation System Project In R [Internet][cited 2021 Nov 1] Available from: https://rpubs.com/ezrasote/movie_recommendation

19. Shuyu Luo, (Dec 10, 2018) Introduction to Recommender System [Internet][cited 2021 Dec 20] Available from: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

20. Jake Moore, (Feb 19, 2020) Python: Implementing Matrix Factorization from Scratch [Internet][cited 2021 Dec 15] Available from: https://towardsdatascience.com/recommender-systems-in-python-from-scratch-643c8fc4f704

21. Emma Grimaldi, (Oct1, 2018) How to build a content-based movie recommender system with Natural Language Processing [Internet][cited 2021 Dec 20] Available from: https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243

22. Abhjit Roy, (Jul 29, 2020) Introduction To Recommender System-1: Content -Based Filtering And Collaborative Filtering [Internet][cited 2021 Oct 20] Available from: https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421

23. Gokul Pisharody, (Now 05, 2020) How to Build a Recommender System (RS) [Internet][cited 2021 Oct 20] Available from: https://inblog.in/How-to-Build-a-Recommender-System-RS-eCv1raie51

24. Saimahdu Palmuri (April 11, 2015) Five most popular similarity measurements implementation in python [Internet][cited 2021 Oct 20] Available from: Five most popular similarity measures implementation in python (dataaspirant.com)

25. Claire Logo (Nov 23, 2018) Evaluation Metrics for Recommender Systems python [Internet][cited 2021 Oct 30] Available from: Evaluation Metrics for Recommender Systems | by Claire Longo | Towards Data Science

26. Denise Chen, (Jul 8, 2020) Recommender System – Matrix Factorization [Internet] Available from: Recommender System — Matrix Factorization | by Denise Chen | Towards Data Science