**Integrated Movie Recommendation System**

**Akash Singh Sparsh Misra Shreesh Srivastava**

**17103171 17103326 17103351**

**Mrs.** **Sherry Garg**

**(Supervisor)**

****

**May – 2020**

**Submitted in partial fulfillment of the Degree of Bachelor of Technology in Computer Science Engineering**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA**

### **TABLE OF CONTENTS**

### **Chapter No. Topics Page No.**

### **Chapter-1 Introduction 9-12**

### 1.1 General Introduction

### 1.2 Problem Statement

### 1.3 Significance/Novelty of the problem

### 1.4 Empirical Study

### 1.5 Comparison of existing approaches to the problem framed

### **Chapter-2** **Literature Survey** 12-23

### 2.1 Summary of papers studied

### 2.2 Integrated summary of the literature studied

### **Chapter 3: Solution Approach** 23-38

### 3.1 Overall description of the project

### 3.2 Requirement Analysis

### 3.5 Solution Approach

### **Chapter-4 Modeling and Implementation Details** 39-42

### 4.1 Design Diagrams

### 4.1.1Use Case diagrams

### 4.1.2 Class diagrams / Control Flow Diagrams

### 4.1.3 Sequence Diagram/Activity diagrams

### 4.2 Implementation details and issues

### 4.3 Risk Analysis and Mitigation

### **Chapter-5 Testing (Focus on Quality of Robustness and Testing)** 43-46

### 5.1 Testing Plan

5.1 Results

### **Chapter-6 Findings, Conclusion, and Future Work** 46-49

### 6.1 Findings

### 6.2 Conclusion

### 6.3 Future Work

### **References** IEEE Format (Listed alphabetically) 49-49

**DECLARATION**

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Name and enrollment number:

Sparsh Misra(17103326)

Akash Singh(17103171)

Shreesh Srivastava(17103351)

Date:

15th May, 2020

**CERTIFICATE**

This is to certify that the work titled “**Integrated Movie Recommendation System**” submitted by “**Shreesh Srivastava, Sparsh Misra, Akash Singh**” in partial fulfillment for the award of the degree of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Name of Supervisor Mrs. Sherry Garg

Designation ASSISTANT PROFESSOR

Date 15 May 2020

**ACKNOWLEDGEMENT**

We are exceptionally obliged to Jaypee Institute Of Information Technology for their direction and consistent supervision and for giving vital data in regards to the undertaking and additionally for their help in finishing the task. We want to offer our thanks towards my folks and individuals from Jaypee Institute Of Information Technology for their kind co-task and consolation which help me in consummation of this venture. We also take this opportunity to express our deepest and sincere gratitude to our supervisor Mrs. Sherry Garg, DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY for her insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

Name of Student Sparsh Misra, Shreesh Srivastava, Akash Singh

Enrollment Number 17103326, 17103351, 17103171

Date 15 May 2020

**SUMMARY**

In this project we will aim to implement a cross-platform recommendation system that takes the data of movies and shows across platforms to give a recommendation system that suggests based on viewed content on other platforms. We do this using cosine similarity, **collaborative filtering** and hybrid recommendation system. This is given an interface providing Flask.

**Name Name of the Supervisor**

Sparsh, Shreesh, Akash Mrs. Sherry Garg

**Date** 15 May, 2020

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table Number** | **Table Heading** | **Page Number** |
| 1 | Paper-1 Information | 12 |
| 2 | Paper-2 Information | 14 |
| 3 | Paper-3 Information | 15 |
| 4 | Paper-4 Information | 17 |
| 5 | Paper-5 Information | 18 |
| 6 | Paper-6 Information | 20 |
| 7 | Paper-7 Information | 22 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure number** | **Figure Heading** | **Page No.** |
| 1 | Fig 1- Basic working of a recommendation system | 39 |
| 2 | Fig 2 Types of recommender system | 40 |
| 3 | Fig 3 Content based filtering | 40 |
| 4 | Fig 4- Item based collaborative filtering | 41 |
| 5 | Fig 5- Hybrid engine | 41 |
| 6 | Fig 6 - Control Flow | 42 |

**Introduction**

## **1.1 General Introduction**

A recommendation system, or a recommendation system (sometimes replaces a 'program' with the same distraction as a platform or engine), is a unique filtration system that seeks to predict the "rating" or "choice" the user can offer to an item. They are used primarily in marketing applications.

Recommendation programs are used in various locations and are often viewed as playlists for video and music services such as Netflix, YouTube, and Spotify, product advisors for services such as Amazon, or content recommendations for social media platforms such as Facebook and Twitter.

Recommendation programs often make use of interactive filtering and content-based filtering (also known as personality-based methodology), and other programs such as information systems. Interactive filtering methods build the model from previous user behavior (previously purchased or selected items and / or price estimates assigned to those items) and similar decisions made by other users. This model is used to predict items (or item ratings) that the user may be interested in.Content-based filtering methods use a series of disc features, marking an item before recommending additional items that have the same properties.

## **1.2 Problem Statement**

## In the spread of information, how to quickly find one’s favorite movie in such a large number of movies becomes a very important issue. Today we have so much content on so many platforms like Netflix, Amazon Prime, AppleTv, HBO, Hotstar, Voot, etc. These platforms have their own recommendation engines for their content, but there is no platform that suggests the content from the various platform altogether. We try to build a cross platform setup to recommend movies from different platform.

## **1.3 Significance of the problem**

We provide a solution to this problem through our integrated movie recommendation system, which collects data from various platforms and provides the user with all the trending, top-rated content and suggestions based on user watch history.

Firestick is the central repository for all your materials. Once logged in, you will have access to any music and videos purchased through your Amazon account. Additionally, you can view any photo you upload to your Amazon Cloud service.

Using your device, you also have access to thousands of apps and games. For example, you can open the Netflix app and use your Fire TV Stick to watch trending Netflix videos. Or, open YouTube and browse the latest uploads from your favorite vlogger. Other services like HBO Now, ESPN and Hulu can also be accessed with your Fire TV Stick at no cost. However, even with a fee, you have the advantage of being able to choose your preferred service without being tied to the cable package.

Chromecast is a streaming media adapter from Google that allows users to play online content such as video and music on digital television.

The adapter is the dongle that plugs into the TV's HDMI port; A cable connects to a USB port to power the device. The mobile app is essentially a smartphone, tablet, laptop or desktop computer that can be used as a TV remote. Once streaming starts, there is no need to keep the app open and the device can be used for other purposes. Chromecast can stream content from multiple sources, including Netflix, Hulu Plus, YouTube, Google Play Music and Movies, and the Chrome browser.

Chromecast competes with other devices for streaming media, including Roku's streaming stick and Apple TV.

## **1.4 Empirical studies**

In the past people used to get one or two online platform subscriptions. These online entertainment platforms have very good content, but even the worst content, like Netflix, has an old library. Trash Films. So, people should also surf on this kind of content.

So, why not an integrated platform where you can get the content you like from multiple platforms in one place?

## 

## **1.5 Comparing current approaches to the problem**

Initial methods include collaborative filtering, which is further enhanced by the use of clustering in combination with K-clustering methods. However, the model has low scalability, data misfolding problems, and high computer sizes at higher order sizes. So slow and awkward recommendations arise.

An attempt was made to reduce the size by using type marks and calibration to obtain the correct algorithm to display the shape. In various clustering algorithms such as k-means algorithm, birch algorithm, mini-batch k-means algorithm, interaction exchange algorithm, distribution algorithm, clustering algorithm and group interaction algorithm. The paper concludes that the best approach is to isolate the Birch algorithm. However, it is not used in models with high quality features.

The following attempts have been made to add the K-Means algorithm. PCA-GAKM is the right solution because it provides the lowest possible error. This is preferred due to low computational cost and data sparsity issues.

One way to use cultural metadata is to suggest the idea of ​​organizing data on the same content based on the liking of similar people. It uses 5 types of content metadata provided by IMDB - user comments, frame frames, summary, plot keywords and styles to get movie recommendations.

# **Literature Survey**

## **Summary**

**Table 1: paper-1 Information**

|  |  |
| --- | --- |
| **Title** | Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm |
| **Author** | Bei-Bei Cui |
| **Publication Detail** | The 4th Annual International Conference on Information Technology and Applications (ITA 2017) |
| **Year** | 05 September 2017 |
| **Summary** | Using the KNN algorithm and collaborative filtering algorithm, it is critical to automatically acquire user-interested films and consumer-interest films in . The key research contents are to help users to obtain user-interested movies automatically in the massive movie information data using the KNN algorithm and collaborative filtering algorithm and to develop a prototype of a movie recommendation system based on the KNN collaborative filtering algorithm.  KNN algorithm is called the K nearest neighbor classification algorithm. The core idea of the KNN algorithm is: if the majority of the k most similar neighbors of the sample in the feature space belongs to a certain category, then the sample is considered to belong to the category.  A collaborative filtering algorithm is categorized as a user-based collaborative filtering algorithm and project-based collaborative filtering. The basic principles of the two are quite similar, and this section mainly introduces the user-based collaborative filtering recommendation algorithm. The basic idea of a collaborative filtering recommendation algorithm is to introduce the information of similar-interest users to object users.  The basic idea of the algorithm is based on records of the history score of the user. Find the neighbor user as u` who has similar interest with target user u, and then recommend the items which the neighbor user u` loved to target user u, the predicted score which targets user u may give on the item is obtained by the score calculation of neighbor user u` on the item. The algorithm consists of three basic steps: user  ulk film data, and develop a prototype of the movie recommendation system based on the KNN collaborative filtering algorithm. The research material is intended to help consumers.  The KNN algorithm is called the K nearest neighbor classification algorithm. The main idea of ​​the KNN algorithm is: If the class of neighboring neighbors of a sample in the feature space belongs to a particular category, then the model is considered to belong to that category.  Collaborative filtering algorithm is classified as user-oriented collaborative filtering algorithm and project-based collaborative filter. The basic principles of both are very similar and this section mainly introduces a user-oriented collaborative filter recommendation algorithm. The basic idea of ​​a collaborative filter recommendation algorithm is to introduce users with single-interest user information for the object.  The basic idea of ​​the algorithm is based on the user history score record. You can find a neighborhood user who is equally interested in the target user, and then recommend the neighborhood user preference, which the user prefers to target, approximately any item. But you can give the user, the score is obtained by counting the user in the neighborhood of the item. The algorithm consists of three basic steps: the user  similarity calculation, nearest neighbor selection, and prediction score calculation. |

**Table 2: paper-2 Information**

|  |  |
| --- | --- |
| **Title** | Design of an Unsupervised Machine Learning-Based Movie Recommender System |
| **Author** | Debby Cintia Ganesha Putri, Jenq-Shiou Leu, Pavel Seda |
| **Publication Detail** | 1  Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei City 106, Taiwan  2  Department of Telecommunications, Brno University of Technology, Technicka 12, 61600 Brno, Czech Republic  3  Institute of Computer Science, Masaryk University, Botanica 554/68A, 602 00 Brno, Czech Republic  \*  Author to whom correspondence should be addressed. |
| **Year** | Received: 25 December 2019 / Revised: 11 January 2020 / Accepted: 13 January 2020 / Published: 21 January 2020 |
| **Summary** | The purpose of this research is to determine equality between groups of people to create a film recommendation system for consumers. As film information has increased, it has become increasingly difficult for consumers to find appropriate films. The recommendation system is very useful in helping customers to select a favorite movie with existing features.  In this study, the advisory system is developed using several algorithms to achieve the clusters such as K-means algorithm, Birch algorithm, Mini-batch K-means algorithm, Average-shift algorithm, Affiliate propagation algorithm, Agrometric clustering algorithm and Spectral clustering. Algorithm. They are proposing methods to optimize Kashmir without increasing the diversity in each cluster. They use groups based on genre and tags for movies are limited. This research could find better ways to evaluate clustering algorithms. |

**Table 3: paper-3 Information**

|  |  |
| --- | --- |
| **Title** | 1. An improved collaborative movie recommendation system using computational intelligence |
| **Author** | Zan Wang, Xue Yu, Nan Feng, Zhenhu Wang |
| **Publication Detail** | [Journal of Visual Languages & Computing](https://www.sciencedirect.com/science/journal/1045926X)  [Volume 25, Issue 6](https://www.sciencedirect.com/science/journal/1045926X/25/6), Pages 667-675 |
| **Year** | December 2014 |
| **Summary** | Using K-means with GA and PCA. Recommended systems have been prevalent in recent years as they address the issue of information overload by suggesting the most relevant products to consumers from the sheer volume of data.  Principal Component Analysis: The main idea of ​​PCA is to convert the original data into a new coordinate space that represents the core of the data with the highest eigenvalue. The first principal component vector provides the most important information after ordering them from high to low levels by eigenvalues. In general, low-importance components are ignored to create positions with less dimensions than the original ones. This reduces computational cost by limiting the dimensionality without compromising the quality of the recommendations.  Since the high-dimensionality similarity of the initially largely empty user-rating matrix makes it difficult to compute, our approach begins with the PCA-based dimension reduction method.  For media products, online collaborative film recommendations help users access their favorite movies by capturing the same neighborhood among users or movies from their historical common ratings. However, due to the data sparsity, neighbor selection is getting more difficult with the fast increase of movies and users.  It employs principal component analysis (PCA) data reduction technique to dense the movie population space which could reduce the computation complexity in intelligent movie recommendation as well. The experiment results on Movielens dataset indicate that the proposed approach can provide high performance in terms of accuracy, and generate more reliable and personalized movie recommendations when compared with the existing methods. |

**Table 4: paper-4 Information**

|  |  |
| --- | --- |
| **Title** | Exploring Movie Recommendation System Using Cultural Metadata |
| **Author** | [Shinhyun Ahn](https://ieeexplore.ieee.org/author/37652903800) ; [Chung-Kon Shi](https://ieeexplore.ieee.org/author/37531102100) |
| **Publication Detail** | [2008 International Conference on Cyberworlds](https://ieeexplore.ieee.org/xpl/conhome/4741259/proceeding) |
| **Year** | Date of Conference: 22-24 Sept. 2008  Date Added to IEEE *Xplore*: 09 January 2009 |
| **Summary** | They developed a simple and low-cost movie recommendation system harnessing vast cultural metadata, about movies, existing on the Web. A simple and low-cost movie recommendation system harnessing vast cultural metadata, about movies, on the Web, and analyzed the strength of the system. As a result, we could be aware of the potential of cultural metadata.  ‘Recommendation systems’ are services which recommend users new items such as news articles, books, music, and movies they would like. They developed a simple and low-cost movie recommendation system harnessing vast cultural metadata, about movies, existing on the Web. Then we evaluated the system, and analyzed its strength. As a result, we could be aware of the potential of cultural metadata. |

**Table 5: paper-5 Information**

|  |  |
| --- | --- |
| **Title** | The YouTube video recommendation system |
| **Author** | James Davidson, Benjamin Liebald, Junning Liu |
| **Publication Detail** | RecSys '10: Proceedings of the fourth ACM conference on Recommender systems |
| **Year** | September 2010 |
| **Summary** | It discusses the use of video recommendation systems on YouTube, the world's most popular online video community. The system recommends a set of personalized videos to users based on their activity on the site. They discuss some of the unique challenges facing the system and how we can address them. In addition, they provide details on experiment and evaluation frameworks to test and tune new algorithms.  Personal recommendations are an important method for information retrieval and content innovation in today's information-rich environment. Combined with pure search (query) and browsing (direct or non-direct), it allows users to encounter massive amounts of information in order to navigate that information efficiently and satisfactorily. As the largest and most popular online video community with user-generated content, YouTube offers some unique opportunities and challenges for content discovery and recommendations.  Users come to YouTube for a variety of reasons, extending the spectrum from minimum to specific: to watch any video they find elsewhere (live navigation), search for specific videos around an item (search and target-oriented browsing), or just be entertained by content they find interesting. Personalized video recommendations are one way to solve this end-use case, which is what we want unofficially.  They choose a batch-oriented pre-computation approach rather than on-demand calculation of recommendations. This has the advantages of allowing the recommendation generation stage access to large amounts of data with ample amounts of CPU resources while at the same time allowing the serving of the pre-generated recommendations to be extremely low latency. The actual implementation of YouTube’s recommendation system can be divided into three main parts: 1) data collection, 2) recommendation generation and 3) recommendation serving. |

**Table 6: paper-6 Information**

|  |  |
| --- | --- |
| **Title** | A framework for diversifying recommendation lists by user interest expansion |
| **Author** | Zhang Zhu, Zheng Xiaolong, Zeng Daniel Dajun |
| **Publication Detail** | [Knowledge-Based Systems](https://www.sciencedirect.com/science/journal/09507051) [Volume 105](https://www.sciencedirect.com/science/journal/09507051/105/supp/C) |
| **Year** | 1 August 2016 |
| **Summary** | Recommended systems are widely used in this age of information overload to search for user preferences and recommend interesting content to customers. Recommendation systems researchers realize that the quality of the Top-N recommendation list is not only ance but also varied.  Most traditional recommendation algorithms make it difficult for each user to create a diversified item list that can cover his or her interests, as they are mainly focused on assessing the exact same aspects as the consumer's dominant interests. In addition, they rarely use semantic information such as item tags and consumer interest labels to improve the type of recommendation. In this paper, we propose a novel recommendation framework that adopts an extension strategy of consumer interests based primarily on social tagging information.  The framework expands the diversity of user preferences by expanding the size and scope of actual user-item interaction records, and then follows the traditional recommendation model for creating recommendation lists. Empirical evaluation on three real-world data sets indicates that our method can effectively improve the accuracy and type of item recommendation. |

**Table 7: paper-7 Information**

|  |  |
| --- | --- |
| **Title** | Distributed collaborative filtering with singular ratings for large scale recommendation |
| **Author** | [Ruzhi Xu, Shuaiqiang Wang, Xuwei Zheng, YinongChen](https://www.sciencedirect.com/science/article/abs/pii/S0164121214001150#!) |
| **Publication Detail** | Journal of Systems and Software [Volume 95](https://www.sciencedirect.com/science/journal/01641212/95/supp/C) |
| **Year** | September 2014 |
| **Summary** | Collaborative filtering (CF) is an effective technique to solve the problem of information overload, where each user associates a rating score with a set of items. For a selected target user, traditional CF algorithms measure similarity between this user and other users, using paired rating scores on common rated items, but only one of them excludes rated scores. We call these comparative scores a dual rating, but we call the comparative scores a single rating. Only 10% of our experiments have dual ratings that can be used for similarity evaluation and the remaining 90% are singular. In this paper, we propose a SingCF approach that seeks to incorporate multiple single ratings, along with dual ratings, to implement a collaborative filter, with the aim of improving the recommended accuracy. We first estimate the score without a score for a single rating and then double it. We perform a CF process to find neighboring customers and create expectations for each target user. Additionally, we provide a MapReduce-based distributed framework in Hadoop for significant improvements in efficiency. Experiments demonstrate the performance advantage of our approach compared to most modern methods. |

# **Overall description of the project and Solution Approach**

**3.1 Description of Algorithm**

Implementing some recommendation algorithms (content-based, popularity-based and collaborative filtering) and trying to create a set of these models to come up with our final recommendation system.

We have two movie datasets:

First dataset: 270,000 users applied for 45,000 images, with 26,000,000 ratings and 750,000 tag applications. The 1,100 tags contain tag genome data with 12 million related scores.

Second dataset: a compilation of 100,000 ratings and 1,300 tag applications for 9,000 movies by 700 users.

The second dataset is used because its size is smaller than the first dataset and we have less computing power.

Showing Trending Films - The basic idea behind this recommendation is that the more popular and critically acclaimed films are more likely to be liked by the average audience. This model does not make individual recommendations based on the user.

The implementation of this model is trivial. All we have to do is sort our movies by rating and popularity and showcase the top movies on our list.

We used IMDB ratings to come up with our top movie chart. I use IMDB's weighted rating formula to build my chart. Mathematically, this is represented as follows:

**Weight Rating (WR) = (v / (v + m) .R) + (m / (v + m) .C)**

Where

**v**- Number of votes per picture

**m**- Minimum votes to be listed on the M chart

**R** is the average rating of the image

**C** - The average vote in the entire report

The next step is to determine the appropriate value for me which is the minimum vote listed on the chart. We use the 95th percentile as our cutoff. In other words, to show an image on a chart, the list must have at least 95% of the votes.

Therefore, considering the chart, the movie in IMDB should have at least 434 votes. We have also seen that the average rating of an image on the TMDB is 5.244 on a scale of 10.44. 2274 films deserve to be on our charts.

Sorting the images according to the value received by the sources and finally displaying the top 15 to 20 movies to the user.

Once we are done with the trending section, we will focus on filtering the images and recommending the user based on the style type. We use the same method to recommend by style, they are the same formula, but when calculating the weight rating we change the metric, that is, we change the cut-off from 0.95 to 0.85.

Those we recommended in the previous section suffer from some serious limitations. For one, it gives the same recommendation to everyone, regardless of the user's personal taste. If someone likes porn movies (and hates action), he'd like to see our Top 15 charts, which movies don't like. If he wants to take a step back and look at our charts across genres, we haven't got the best recommendations yet.

So now we do content based filtering to represent images to suit the users' tastes.

To further personalize your recommendations, I'm going to create an engine that calculates the similarity between images based on certain metrics and refers to movies that are similar to the specific image the user prefers, also known as content-based filtering (or content) because the content uses movie metadata.

Due to the low computing power we have to reduce the size of the dataset.

Then we recommend using a description of the image.

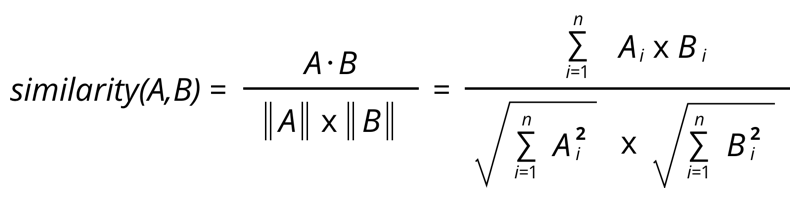
The two most common types of recommended systems are content-based and collaborative filtering (CF).

Collaborative filtering generates recommendations based on the user's knowledge on the attitude of the item, which uses "Crowd Knowledge" to recommend the items.

Content-based recommendation systems focus on the properties of objects and give you recommendations based on the similarity between them.

### Cosine Similarity

We will be using the Cosine Similarity to calculate a numeric quantity that denotes the similarity between the two movies. Mathematically, it is defined as follows:



Since we've got used the TF-IDF Vectorizer, calculating the scalar product will directly give us the Cosine Similarity Score. Therefore, we'll use sklearn's linear\_kernel rather than cosine\_similarities since it's much faster.

We will have a pairwise cosine similarity matrix for all the flicks in our dataset.

Unfortunately, this can be not of much use to the general public because it doesn't take into consideration important features like cast, crew, director, and genre, which determine the rating and also the popularity of a movie. we'll build a more sophisticated recommender that takes genre, keywords, cast, and crew into consideration.

### 

### Metadata Based Recommender

To create our standard metadata-based content recommendation, we must combine our existing dataset with staff and keyword datasets.

Of the staff, we only choose the director as our facilitator because the others don't contribute much to the spirit of the film.

Choosing a cast may be a bit difficult. Less well-known actors and minor characters don't really affect people's opinions on a movie. Therefore, we only need to choose the most characters and their cast. We unilaterally select the highest 3 actors that appear on the credit list.

Creating a metadata dump for every image containing genres, directors, main actors, and keywords. Countvectorizer provides a straightforward thanks to tokenize a set of text documents and make a glossary of known words, but also to encode new documents using that terminology.

The rest of the steps are kind of like what we've done before: we treat cosmic similarities and return films very similarly.

We do a minimal amount of processing before using any of our keywords. As a primary step, we calculate the frequency and count of every keyword found within the dataset. Now, it's like an evidence recommendation.

We have no use for keywords that occur just the once. this suggests that the descriptor dataset must have some v image, which implies it's often used, that the frequency of the word must be a minimum of once. Therefore, they'll be safely removed. Finally, we alter each word in its trunk in order that words like dog and dog behave the identical.

One thing we notice about our recommendation system is that it recommends movies irrespective of rating and recognition. Therefore, we'll add a mechanism to get rid of bad images and return popular and good critical response.

We use the primary recommendation of flicks with a replacement metric supported the new similarity score, which implies the cut-off value is 0.6. Then, using this because the m value, we calculate the weighted rating of every image using the IMDB formula as we did within the General Recommendations section.

Just because they need the identical character or style doesn't mean that the user will prefer the recommended movies. Therefore, we recommend employing a collaborative filter supported popularity and ratings.

### Memory based Collaborative Filtering

A *user-item filtering* will take a particular user, find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked.In contrast, *item-item filtering* will take an item, find users who liked that item, and find other items that those users or similar users also liked. It takes items and outputs other items as recommendations.

*Item-Item Collaborative Filtering*: “Users who liked this item also liked …”

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

In both cases, we create a user-item matrix which is built from the entire dataset.

After we have built the user-item matrix you calculate the similarity and create a similarity matrix. The similarity values between items in *Item-Item Collaborative Filtering* are measured by observing all the users who have rated both items.

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

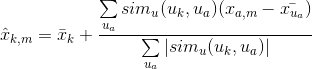
A distance metric commonly used in recommender systems is *cosine similarity*, where the ratings are seen as vectors in n-dimensional space and the similarity is calculated based on the angle between these vectors. Cosine similarity for users *a\* and \*m* can be calculated using the formula below, where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



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



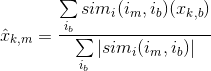
The next step is to make predictions. You have already created similarity matrices: user\_similarity and item\_similarity and therefore you can make a prediction by applying the following formula for user-based CF:



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

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

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



### Model Based Collaborative Filtering

The model-based collaborative filtering is based on the matrix factor (MF), which has gained maximum exposure, mainly as a non-practicing learning method for latent variable decomposition and dimensionality reduction. Matrix factorization is more widely used to recommend systems that can deal with scalability and sparsity than memory-based CFs.

The goal of MF is to learn the users' hidden preferences and learn the hidden characteristics of objects from known ratings (know the characteristics that describe the ratings) and then learn the hidden characteristics of users and goods through a dot product of unknown ratings. Let us take it. When you have many dimensions, with many dimensions, you can reorganize the user-item matrix into a low-rank structure by multiplying the matrix factor, and you can represent the matrix by multiplying two lower-level matrices. The rows contain the latent vector. This matrix is ​​enough for you to estimate your original matrix by multiplying the low-rank matrix as much as possible, which fills out the missing entries in the original matrix.

Models that use both rating and content characteristics are called hybrid recommendation systems, where both collaborative filtering and content-based models are combined. Hybrid recommendation systems usually show greater accuracy than collaborative filtering or content-based models: they can solve the cold start problem because you don't have a user rating or you can use metadata. From the user or item to be assessed.

### 

### SVD

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

Given m x n matrix X:

* *U* is an *(m x r)* orthogonal matrix
* *S* is an *(r x r)* diagonal matrix with non-negative real numbers on the diagonal
* *V^T* is an *(r x n)* orthogonal matrix

Elements on the diagonal in S are known as *singular values of X*.

Matrix *X* can be factorized to *U*, *S* and *V*. The *U* matrix represents the feature vectors corresponding to the users in the hidden feature space and the *V* matrix represents the feature vectors corresponding to the items in the hidden feature space.

Now we can make a prediction by taking the dot product of *U*, *S* and *V^T*.

Will not be implementing Collaborative Filtering from scratch. Instead, I will use the Surprise library that used extremely powerful algorithms like Singular Value Decomposition (SVD) to minimize RMSE (Root Mean Square Error) and give great recommendations.

### Genre-based Recommendation(on Amazon dataset)

The Amazon Prime(India) movies and TV Shows dataset used in this project is not user-based so we can not apply the traditional user-based recommendations on this one, so we have applied content-based recommendations on the basis of genre(mainly). The movies/TV Shows recommended will be similar in Genre, Year of release, IMDb Ratings, and hence it will give an idea how similar a test movie is to all the movies in the list.

The method used in this recommendation is very simple and customized. NO Machine Learning algorithm is used in this type of recommendation engine and it is made purely on the basis of Data Analysis and yet the results are satisfactory.

The dataset consists of genre tags such as ‘Sci-fi’,’Drama’,’Romance’,’Comedy’ etc. and each movie has different tags of genres for eg. the movie ‘Ae Dil hai Mushkil’ has the tags ‘Bollywood’,’Drama’,’Romance’,’ World Cinema’. All the movies sharing the same or a maximum number of tags with this movie have the probability of being similar to this movie. Also, we have added some more features to refine the recommendations such as minimum IMDb Rating, same age rating, near about year of release, etc.

In the end this recommendation system will be integrated with the rest of the recommendation engines.

### Content-Based Recommender:

### A content based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate

We built two content based engines; one that took movie overview and taglines as input and the other which took metadata such as cast, crew, genre and keywords to come up with predictions. We also made a simple filter to give greater preference to movies with more votes and higher ratings.

### Hybrid Recommendation Engine

Most recommender systems now use a hybrid approach, combining [collaborative filtering](https://en.wikipedia.org/wiki/Collaborative_filtering), content-based filtering, and other approaches . There is no reason why several different techniques of the same type could not be hybridized. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model for a complete review of recommender systems). Several studies that empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrated that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem, as well as the knowledge engineering bottleneck in [knowledge-based](https://en.wikipedia.org/wiki/Knowledge_base) approaches.

We brought together ideas from content and collaborative filtering to build an engine that gave movie suggestions to a particular user based on the estimated ratings that it had internally calculated for that user.

We have built a simple hybrid recommender that brings together techniques we have implemented in the content based and collaborative filter based engines. This is how it will work:

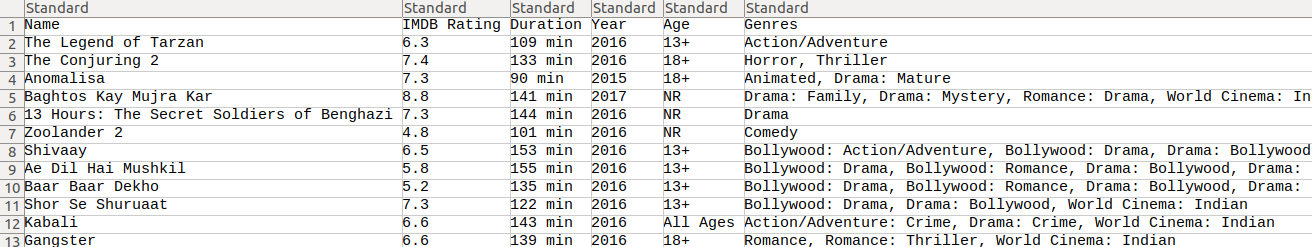
Input: User ID and the Title of a Movie

Output: Similar movies sorted on the basis of expected ratings by that particular user.

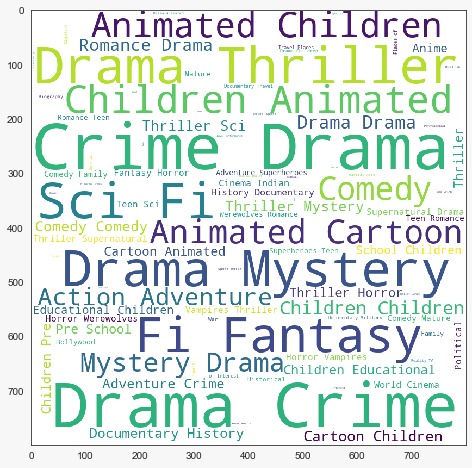
# **Datasets Used**

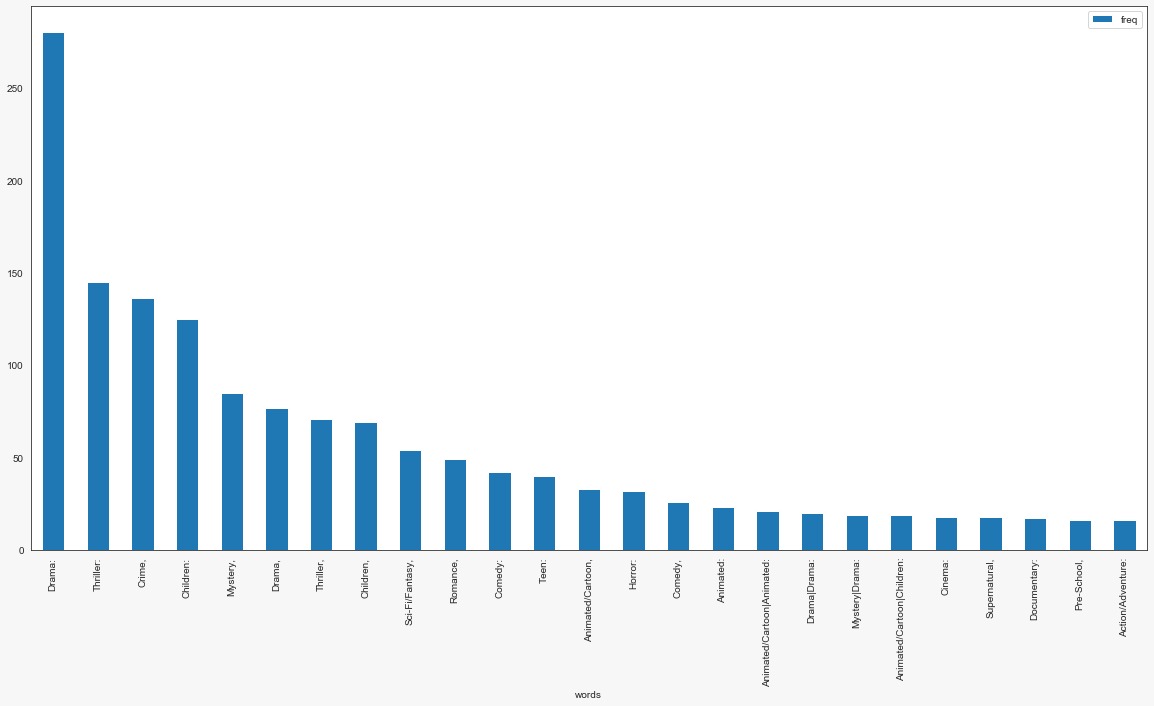
### Amazon Prime Movies and TV Shows Dataset

This Dataset is extracted by web scraping from ‘finder.com’ . Below here is a sample of the dataset. It consists of genre tags which is the main basis of recommendation here.



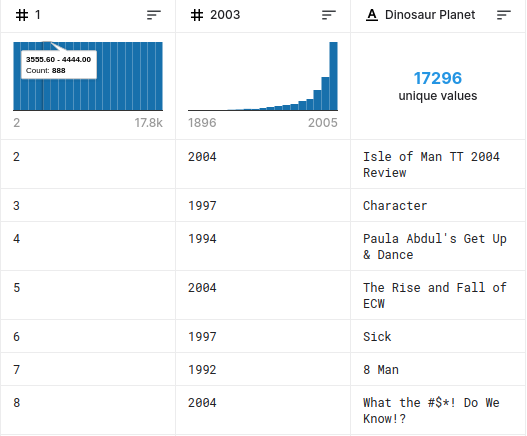
**Genre Visualization**

****

****

### Netflix Dataset

The dataset used here comes directly from Netflix. It consists of 4 text data files, each file contains over 20M rows, i.e. over 4K movies and 400K customers. All together **over 17K movies** and **500K+ customers**!

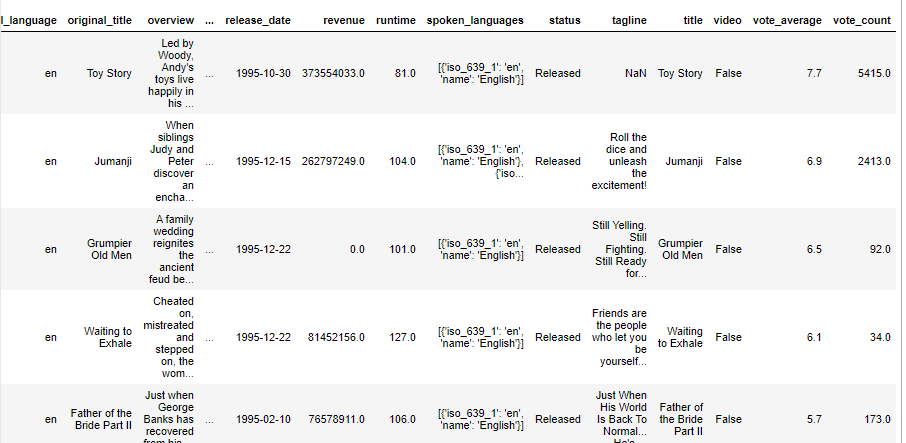


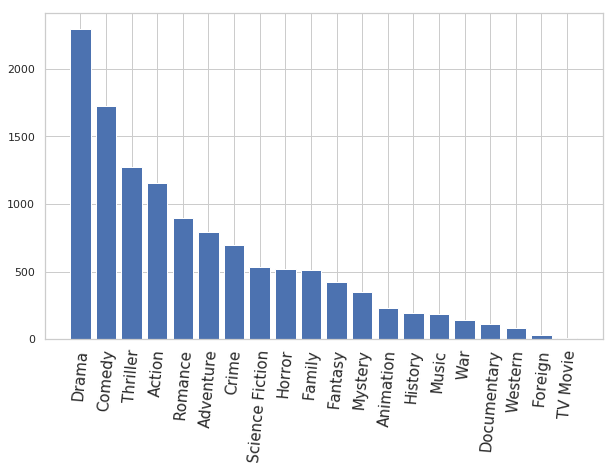
### The Movies Database

These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

This dataset is extracted partly from the Full MovieLens dataset and partly from the small MovieLens dataset.





## **Solution Approach**

Making a separate Python Virtual Environment to work on the linking of our model and the web application.

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. This is one of the most important tools that most of the Python developers use. It is generally good to have one new virtual environment for every Python based project you work on. So the dependencies of every project are isolated from the system and each other.

### **3.3 Requirement Analysis**

We’ve used Flask as our web framework , we’ve tried linking it with our model using a separate python virtual environment.

We’ve used HTML and CSS for making the frontend UI and CSS for styling. Saved all the files in a template of our Flask project folder.

### **Tools and Libraries Used**

1. Scipy: To implement CSR(Compressed Sparse Row Matrix) for its advantage in efficient arithmetic operations CSR + CSR, CSR \* CSR, etc, efficient row slicing and fast matrix vector products.
2. NLTK: The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.
3. Surprise: Which is a python scikit to implement recommender systems. It provides tools to evaluate, analyse and compare the algorithms performance. Cross-validation procedures can be run very easily using powerful CV iterators (inspired by scikit-learn excellent tools), as well as exhaustive search over a set of parameters.
4. Seaborn: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
5. Matplotlib: Matplotlib is another powerful Python data visualization library used for all kind of plots. Here we have used it for bar graphs and word clouds.
6. Sci-kit learn : A very important library which contains important packages such as train-test split, TfidfVectorizer, CountVectorizer,linear\_kernel, cosine\_similarity etc.
7. Pandas : Another essential library for importing data, EDA, Dataframes and Series.

## **Modeling and Implementation Details**

### **4.1 Design Diagrams**

Fig 1- Basic working of a recommendation system:

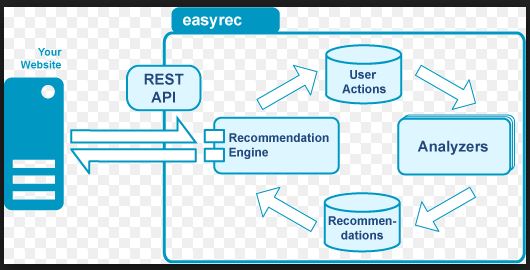


Fig 2 Types of recommender system

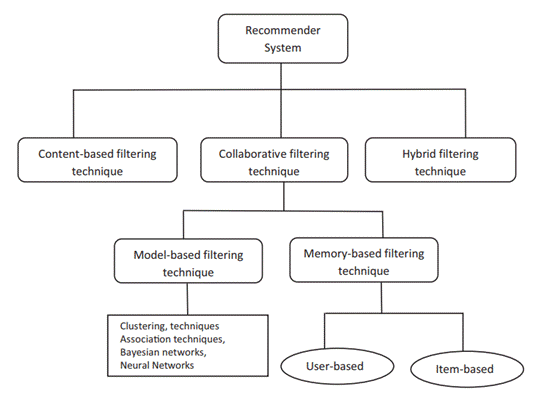


Fig 3 Content based filtering

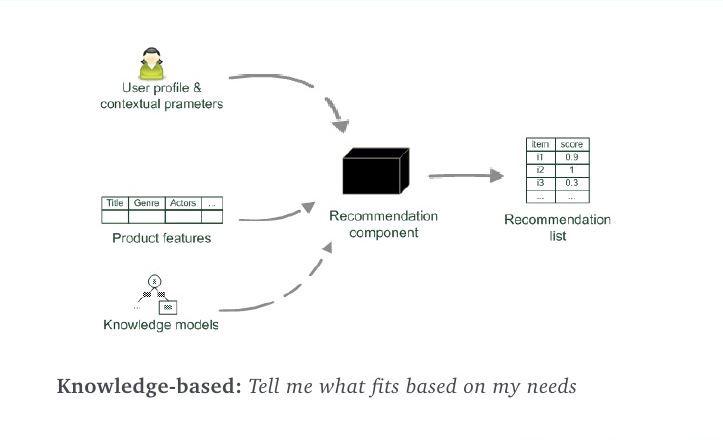


Fig 4- Item based collaborative filtering



Fig 5- Hybrid engine

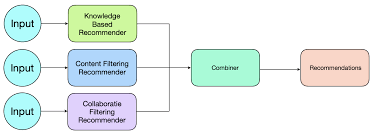
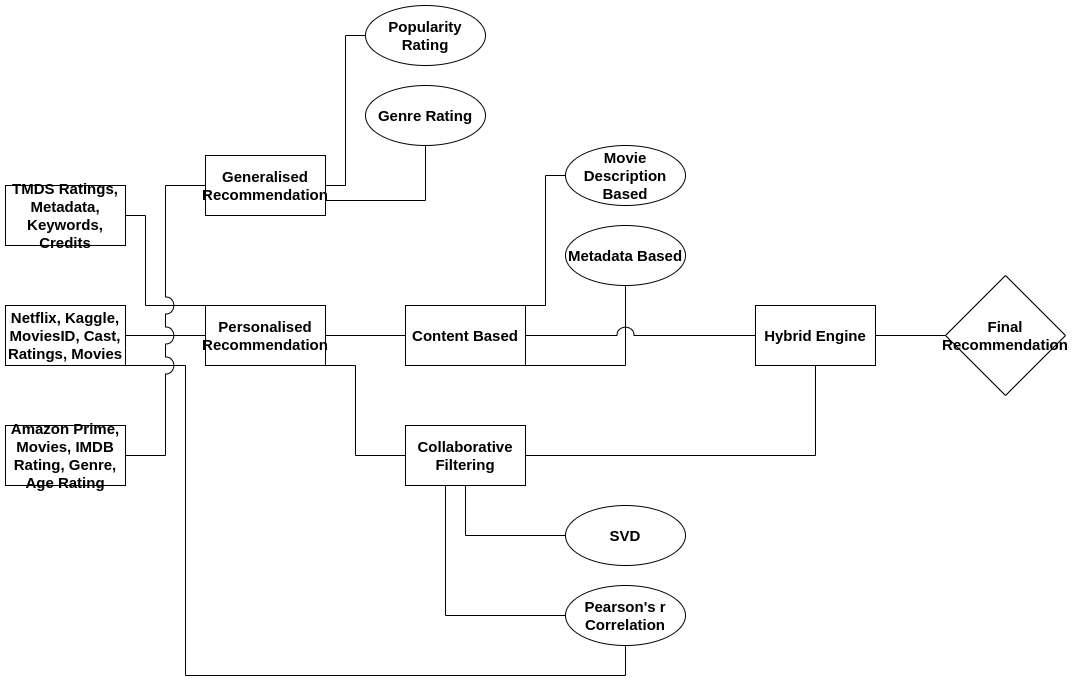
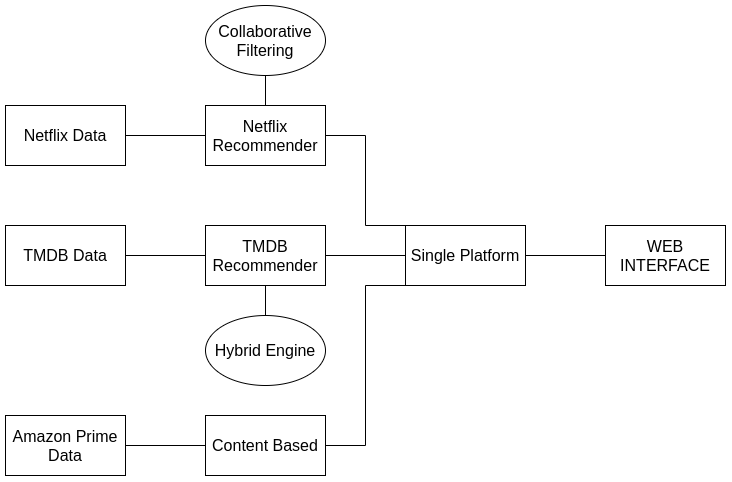


Fig 6 - Control Flow





# **Testing**

## **5.1 Testing Various Algorithms**

Testing different algorithms from simple similarity based to genre and meta data based to collaborative filtering and eventually using the hybrid filtering.

### Cosine Similarity

Since we've used the TF-IDF Vectorizer, calculating the scalar product will directly give us the Cosine Similarity Score. Therefore, we'll use sklearn's linear\_kernel rather than cosine\_similarities since it's much faster.

We will have a pairwise cosine similarity matrix for all the films in our dataset.

Unfortunately, this is often not of much use to most of the people because it doesn't take into consideration vital features like cast, crew, director, and genre, which determine the rating and therefore the popularity of a movie. we'll build a more sophisticated recommender that takes genre, keywords, cast, and crew into consideration.

### Metadata Based Recommender

To build our standard metadata-based content recommender, we'll got to merge our current dataset with the crew and therefore the keyword datasets.

Creating a metadata dump for each movie which consists of genres, director, main actors, and keywords. The CountVectorizer provides an easy thanks to both tokenize a set of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

The remaining steps are almost like what we did earlier: we calculate the cosine similarities and return movies that are most similar.

We will do alittle amount of pre-processing of our keywords before putting them to any use. As a primary step, we calculate the frequency, counts of each keyword that appears within the dataset. Now, this is often sort of a Description recommender.

We don't have any use for keywords that occur just one occasion . meaning the outline should have any relevance within the dataset i.e. it's used often, therefore the word frequency should be a minimum of quite once. Therefore, these are often safely removed. Finally, we'll convert every word to its stem in order that words like Dogs and Dog are considered an equivalent .

One thing that we notice about our recommendation system is that it recommends movies no matter ratings and recognition . Therefore, we'll add a mechanism to get rid of bad movies and return movies which are popular and have had an honest critical response.

Having an equivalent characters or genre doesn’t mean that the user would really like the recommended movies. Therefore, we'll make a recommendation supported popularity and ratings using Collaborative Filtering.

A user-item filtering will take a specific user, find users that are almost like that user supported similarity of ratings, and recommend items that those similar users liked.In contrast, item-item filtering will take an item, find users who liked that item, and find other items that those users or similar users also liked.

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

The goal of MF is to find out the latent preferences of users and therefore the latent attributes of things from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the scalar product of the latent features of users and items. once you have a really sparse matrix, with tons of dimensions, by doing matrix factorization you'll restructure the user-item matrix into a low-rank structure, and you'll represent the matrix by the multiplication of two low-rank matrices, where the rows contain the latent vector. You fit this matrix to approximate your original matrix, as closely as possible, by multiplying the low-rank matrices together, which fills within the entries missing within the original matrix.

### Collaborative Filtering

Our content based engine suffers from some severe limitations. it's only capable of suggesting movies which are on the brink of a particular movie. That is, it's unable to capture tastes and provide recommendations across genres.

Also, the engine that we built isn't really personal therein it doesn't capture the private tastes and biases of a user. Anyone querying our engine for recommendations supporting a movie will receive an equivalent recommendation for that movie, no matter who s/he is.

Therefore, during this section, we'll use a way called Collaborative Filtering to form recommendations to Movie Watchers. Collaborative Filtering is predicated on the thought that users almost like me often want to predict what proportion I will be able to sort of a particular product or service those users have used/experienced but I even have not.

A well-known matrix factorization method is Singular value decomposition (SVD). Collaborative Filtering are often formulated by approximating a matrix X by using singular value decomposition. the overall equation are often expressed as follows:

Elements on the diagonal in S are referred to as singular values of X.

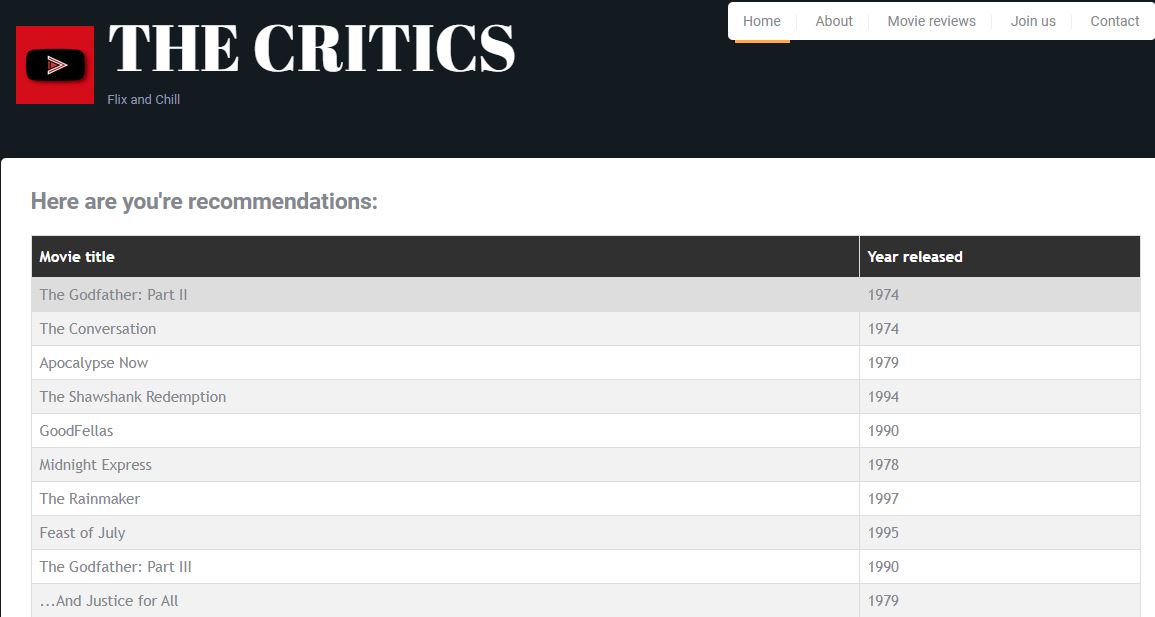
Matrix X are often factorized to U, S and V. The U matrix represents the feature vectors like the users within the hidden feature space and therefore the V matrix represents the feature vectors like the things within the hidden feature space.

Now we will make a prediction by taking the scalar product of U, S and V^T.

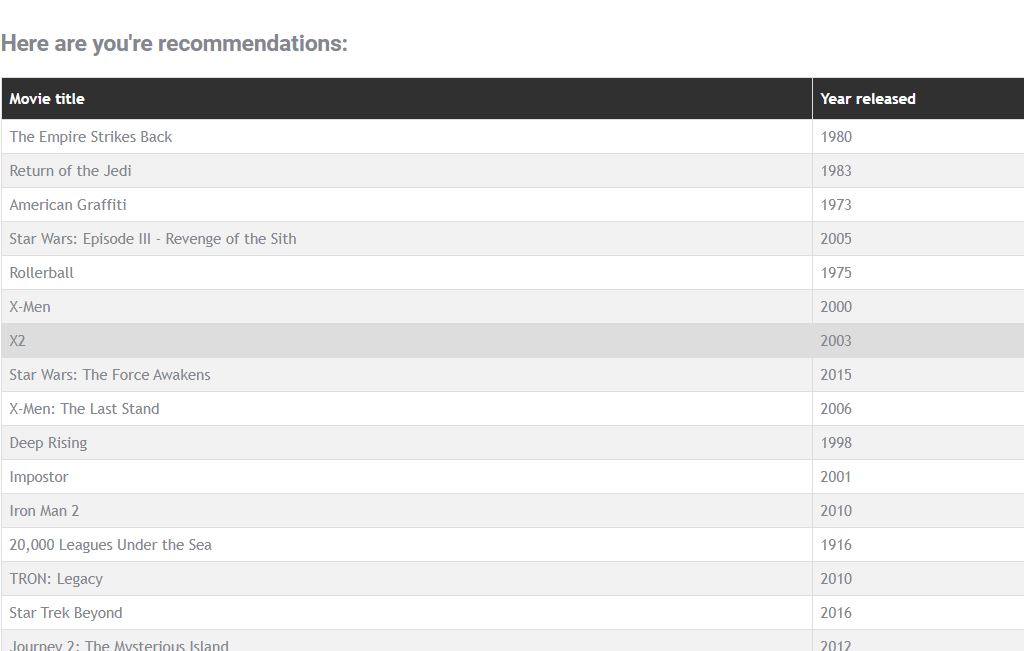
Will not be implementing Collaborative Filtering from scratch. Instead, I will be able to use the Surprise library that used extremely powerful algorithms like Singular Value Decomposition (SVD) to minimise RMSE (Root Mean Square Error) and provide great recommendations.

## **5.2 Results**

Following are the results of the search on our locally hosted website :



**Recommendations for user who liked movies like The Godfather**



**Recommendations for user who liked movies like Star Wars**

# **Findings, Conclusion, and Future Work**

## **6.1 Conclusion**

Learn from data and recommend best TV shows to users, based on self & others behavior. We have achieved a cross-platform recommendation system that can suggest shows and movies on the other platforms using the data of viewing habits collected from another platform.

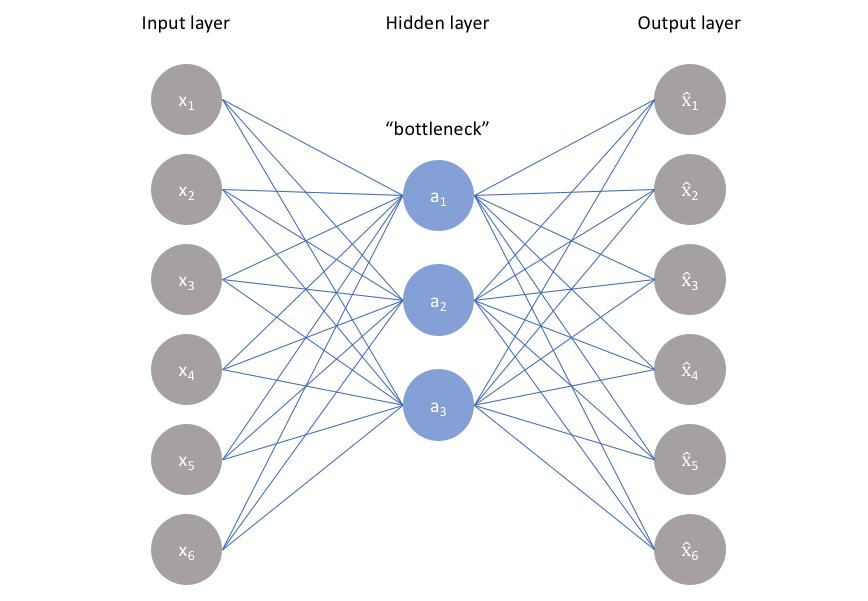
In this notebook, we've built 4 different recommendation engines supporting different ideas and algorithms. they're as follows:

* Simple Recommender: this technique used overall TMDB Vote Count and Vote Averages to create Top Movies Charts, generally and for a selected genre. The IMDB Weighted scoring system was wont to calculate ratings on which the sorting was finally performed. This is often also utilized in the Amazon Prime dataset to classify movies supported genres , similar year of release and IMDb ratings.
* Content Based Recommender: We built two content based engines; one that took movie overview and taglines as input and therefore the other which took metadata like cast, crew, genre and keywords to return up with predictions. We also made an easy filter to offer greater preference to movies with more votes and better ratings. We then used cosine similarity to seek out similarity between movies.
* Collaborative Filtering: In TMDb recommender system, we used the powerful Surprise Library to create a collaborative filter supported single value decomposition (SVD). The RMSE obtained was but 1 and therefore the engine gave estimated ratings for a given user and movie. In Netflix dataset we used the Pearson's R coefficient method to seek out the linear correlation between review many all pairs of flicks , then we offer the highest 10 movies with highest correlations
* Hybrid Engine: We brought together ideas from content and collaborative filtering to create an engine that gave movie suggestions to a specific user supported the estimated ratings that it had internally calculated for that user.

Learn from data and recommend best TV shows to users, supported self & others behavior. we've achieved a cross-platform recommendation system which will suggest shows and films on the opposite platforms using the info of viewing habits collected from another platform.

## **6.2 Future Work**

We can use Autoencoders for dimensionality reduction. Autoencoders are neural networks which will be wont to reduce the info into a coffee dimensional latent space by stacking multiple non-linear transformations(layers). they need an encoder-decoder architecture. The encoder maps the input to latent space and therefore the decoder reconstructs the input. they're trained using backpropagation for accurate reconstruction of the input. within the latent space has lower dimensions than the input, autoencoders are often used for dimensionality reduction. By intuition, these low dimensional latent variables should encode most vital features of the input since they're capable of reconstructing it. Auto-encoders are capable of modelling complex nonlinear functions.



Autoencoders are an unsupervised learning technique during which we leverage neural networks for the task of representation learning. Specifically, we'll design a neural specification such that we impose a bottleneck within the network which forces a compressed knowledge representation of the first input. If the input features were each independent of 1 another, this compression and subsequent reconstruction would be a really difficult task. However, if some kind of structure exists within the data (ie. correlations between input features), this structure are often learned and consequently leveraged when forcing the input through the network's bottleneck.

# **References**

1. Bei-Bei Cui (05 September 2017) Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. In The 4th Annual International Conference on Information Technology and Applications (ITA 2017)
2. Debby Cintia Ganesha Putri, Jenq-Shiou Leu, Pavel Seda ( 21 January 2020 ) Design of an Unsupervised Machine Learning-Based Movie Recommender System. In the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei City 106, Taiwan. In the Department of Telecommunications, Brno University of Technology, Technicka 12, 61600 Brno, Czech Republic. In Institute of Computer Science, Masaryk University, Botanica 554/68A, 602 00 Brno, Czech Republic.
3. Zan Wang, Xue Yu, Nan Feng & Zhenhu Wang (December 2014) An improved collaborative movie recommendation system using computational intelligence. In Journal of Visual Languages & Computing Volume 25, Issue 6, Pages 667-675.e
4. Shinhyun Ahn & Chung-Kon Shi (Date Added to IEEE *Xplore*: 09 January 2009) Exploring Movie Recommendation System Using Cultural Metadata. In 2008 International Conference on Cyberworlds.
5. James Davidson, Benjamin Liebald, Junning Liu (September 2010) The YouTube video recommendation system. In RecSys '10: Proceedings of the fourth ACM conference on Recommender systems.
6. Zhang Zhu, Zheng Xiaolong, Zeng Daniel Dajun (1 August 2016) A framework for diversifying recommendation lists by user interest expansion. In [Knowledge-Based Systems](https://www.sciencedirect.com/science/journal/09507051) (Volume 105).
7. Ruzhi Xu, Shuaiqiang Wang, Xuwei Zheng & Yinong Chen (September 2014) Distributed collaborative filtering with singular ratings for large scale recommendation. In Journal of Systems and Software (Volume 95).
8. Collaborative Filltering (<https://realpython.com/build-recommendation-engine-collaborative-filtering/>)