iSTEAM Communications ISSN:2691-7300 www.iSTEAMC.com

Research Article in Artificial Intelligence

Wingman: A New Movie Recommendation System Algorithm

Aayush Singh[1*]

[1] Guru Gobind Singh Indraprastha University, New Delhi, Delhi 110032, India

[*] Correspondence: aayush.singh2708@gmail.com

(Received: 08/17/2022; Accepted: 09/27/2022; Published: 10/10/2022)

DOI: https://doi.org/10.37906/isteamc.2022.3

Abstract: In this rising new era, the world is growing faster and so are we. People nowadays need effortless services, and recommendation system is one of them. They are present and used everywhere, whether it's advertising, e-commerce or any other domain. Companies such as Netflix, YouTube, Amazon, Spotify rely on recommendation systems to boost their sales. Moreover, it increases the engagement of users with a particular app or website as it tends to show more relevant individualized content. It basically takes the user's past history as an input and then using filtering techniques it predicts the most suitable item/content for the user as an output. The main objective of this paper is to predict movies most desired by the user with as much accuracy as possible. In this paper, I will provide a brief description of various filtering methods and the particular techniques that I used to achieve the desired result. I have used Tf-Idf to quantify the overview of the movies, then did a pairwise similarity for the Content Based Filtering. This allows the user to watch movies based on their watch history according to the plot of the movies. For Collaborative Filtering, I have used Matrix Factorization to generate real-time recommendations based on user-movie rating matrix. Correlation Coefficient is used to calculate the score. The higher the score, the more likely the movie is liked by the user. I have used the voting system for Hybrid Filtering. Weighted Rating Score is calculated with the help of the votes given to the movies by the users. Then a popularity feature is also used to generate more meaningful movie recommendations. MinMax Scalar is used to normalize the data, and then fifty percent priority is given to both Weighted Rating Score and popularity and a final score is calculated. Using this final score, popular or top movies are generated. The databases used are retrieved from MovieLens and Kaggle. Consequently, my proposed system will generate recommendations based on the plot of the movie using tf-idf and cosine similarity, based on user-movie rating matrix and correlation coefficient, based on voting and popularity basis.

Keywords: Recommendation System, Content Based Filtering, Collaborative Filtering, Hybrid Filtering, Term Frequency-Inverse Document Frequency (Tf-Idf), Singular Value Decomposition, Weighted Rating Score

1. Introduction

Steve Jobs once said, "People don't know what they want until you show it to them." (AltexSoft, 2021). We cannot argue this statement as one of the most fascinating systems which is driving the economy at present is the Recommendation System. YouTube, Netflix, Spotify, Amazon, and so on, all these huge companies are employing this system on their particular platforms to heighten up their sales, to attract new customers, and moreover to engage the customers more in their particular platforms. However, all

companies have their separate recommendation systems, and they all are based on various distinct algorithms and different filtering methods.

In this paper, we are going to talk about the need for recommendation systems, what future they hold for us and some other previous Recommendation Models which have been already proposed. Furthermore, we will be discussing the various Filtering Methods: Content Based Filtering, Collaborative Filtering, Hybrid Filtering, and the particular techniques and algorithms we used to generate movie recommendations. Moreover, methodology for every filtering method is given in this paper for better understanding. Following this the simulation results are shown and discussed in the Discussion section. On top of that, various future scopes are also illustrated that can be made in the project proposed in this paper. Finally, the Conclusion for the proposed system is discussed. The main approach of this paper is to generate movie recommendations using first- and second-generation techniques with as much accuracy as possible.

1.1 Need of Study

Here are some motivating purposes for why I started this particular field of study and why this study is crucial in upcoming days:

Movie recommender systems enhance the watching experience of users. Personalized recommendations tend to help the user to engage more on a particular platform. It helps to redefine the user experience. And because of this, it eventually retains the users to that particular platform.

Recommendation engines boost up the economy, by boosting up the speed of searches and making it convenient for users to acquire the content they are most engrossed in. It also helps the user to save up their time and efforts. They try to surprise the users with diverse content which they may adore and which the users would have never even searched for.

Machine learning has become a hot topic, and there is so much to prospect in the field of recommendation systems. Recommendation systems are used in every domain such as to recommend movies, e-products, advertisements, etc. There have been many successful algorithms to generate recommendations. The researchers are trying to make a system more sophisticated and robust that can actually recommend things based on the user's behavior and rationality with much more accuracy. Therefore, this study will help in advancing towards more robust and sophisticated models of recommendation systems.

From an economic perspective, it has proven to elevate the sales and profits for various business platforms in every domain by increasing the interests of users in their particular platforms.

1.2 Scope of Study

A recommendation system is a software filtration program whose main agenda is to help the users to find more appropriate content of their liking in any domain without much effort. In this paper, the domain specific item is a movie, hence this paper will focus on finding appropriate movie recommendations based on the user's history.

- This study focuses towards creating a movie recommendation system that will reduce the time

and efforts of a user to choose a movie of their liking.

- This movie recommendation system consists of three machine learning methods which are: Content Based Filtering, Collaborative Filtering and Weighted Hybrid Filtering.
- This system is using two databases which are: tmdb 5000 movie dataset and a movie lens dataset.

1.3 Literature Review

Luis M Campos et al. studied Content Based and Collaborative Filtering and then proposed a fusion of Bayesian Network and Collaborative Filtering to reduce the drawbacks of Content Based Filtering and Collaborative Filtering (Campos, L. M. d., Fernández-Luna, J. M., Huete, J. F., & Rueda Morales, M. A., 2010).

Urszula Kużelewska et al. proposed clustering as a way to deal with recommender systems. System used two techniques for computing cluster representatives. Memory-based Filtering and Centroid-Based Solution were used to compare effectiveness of the two techniques. Accuracy was increased when compared to just centroid-based methods (Kużelewska, 2014).

"MOVREC" a movie recommendation system introduced by D.K. Yadav et al. which works on collaborative filtering. The data is analyzed and filtrations are made using a collaborative approach to generate movie recommendations based on the movie with highest rating first (Kumar, M., Yadav, D.K., Singh, A., & Gupta, V. K., 2015).

Clusters of user and item specific information are formed using chameleon by Utkarsh Gupta et al. It is based on hierarchical clustering. Voting system is used to predict the ratings of an item. This system has lower error and a finer clustering of similar items (Gupta, U., & Patil, N., 2015, July 13).

Costin-Gabriel Chiru et al. proposed a Movie Recommendation System which uses both Content Based and Collaborative Filtering. The system uses the user's watching history and the data involving movie score from websites is collected. Recommendations are generated using aggregate similarity calculation. It attempts to solve the problem of generating more unique and personalized recommendations for specific users (Chiru, C.-G., Dinu, V.-N., Preda, C., & Macri, M, 2015).

"Recombee", a company based in Prague, has developed a recommendation engine capable of generating recommendations in multiple domains. They have used Multi-Objective Optimization techniques, Deep Learning to generate more accurate and real-time recommendations. Furthermore, advanced Natural Language Processing techniques are combined to reduce the cold start problem (Kordík, 2019).

Hongli Lin et al. proposed a method known as Contentboosted Collaborative Filtering (CBCF). This algorithm is broken into two parts, first one is the content-based filtering that improves the existing trainee case ratings and the second one is the collaborative filtering that provides the predictions. This method tends to include the merits of both CF and CBF, while at the same time, overcoming their demerits (Lin et al., 2014).

2. Research Design

A movie recommendation system is proposed in this paper. The system will use recommendations generated from all the three methods i.e. Content Based Filtering, Collaborative Filtering, Hybrid Filtering. The main objective of the proposed system is to recommend movies with as much precision as possible based on user preferences.

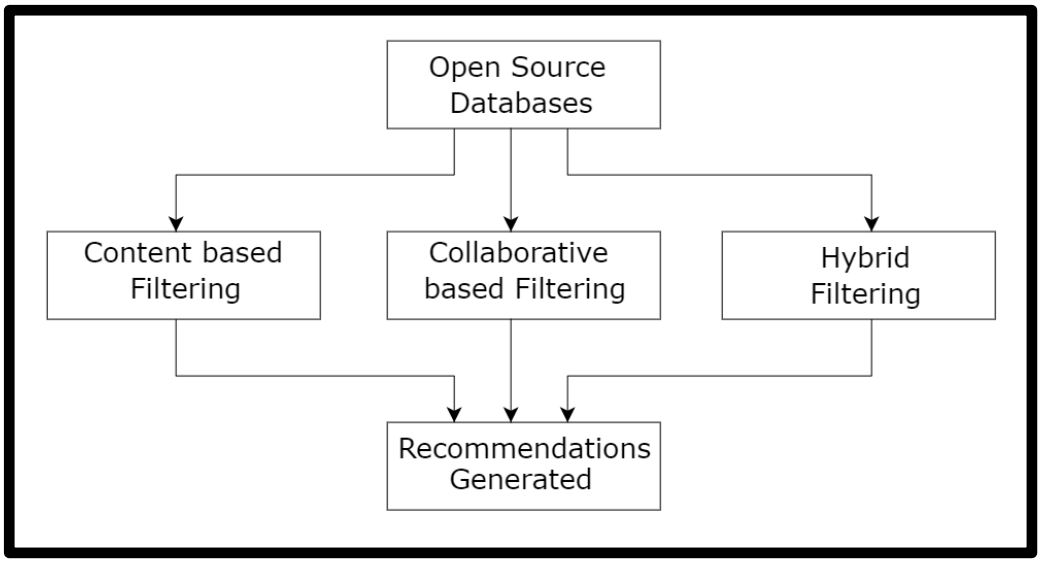

Figure 1: Proposed Movie Recommendation System

2.1 Study 1: Content Based Filtering using Tf-Idf

Content based filtering works on the idea of similarity (Seif, 2019). It basically uses specific item or user attributes and then it finds the similarity between them. After finding the similarity it recommends the most relevant content based on the user's past history. The attributes for user preferences can be age, gender, and personal other factors. It can be achieved through methods such as Tf-Idf, Vector Space Models, cosine similarity, etc. Now, we will focus our discussion on the Tf-Idf method for Content Based Filtering.

This method is used to quantify words in a corpus (Scott, 2019). It can be broken down into two parts: Term Frequency- It works by looking at the number of terms that we are looking for in a corpus.

$$df(t) = count\ of\ t\ in\ d/number\ of\ words\ in\ d$$

Where, t-term, d-document (set of words) and N- count of corpus

Inverse Document Frequency- It basically calculates the informativeness of a word.

$$idf(t) = N/df$$

$$idf(t) = log(N/(df + 1))$$

Log is used to dampen the effect of exploding of the idf value.

Combining these we get inversely related to its frequency across documents (Scott, 2019).

$$tfidf(t,d) = tf(t,d) \times log(N/(df + 1))$$

2.1.1 Design of Study 1:

In this paper, we have first analyzed and preprocessed the data. Then, we did a pairwise similarity to generate the system's recommendations. To achieve this, first we created a vector-matrix using Tf-Idf of the "overview" feature present in the database. This "overview" feature has information about the plot of the movie. Based on this information, we did a pairwise similarity using cosine similarity. We calculated a pairwise similarity score for each movie. The Sigmoid Kernel function is used to calculate the cosine similarity score. The higher the cosine similarity score between two movies, the more they are similar to each other. The movies with high cosine similarity score will be recommended first. The dataset used has the description of the plot of the movie in the "overview" feature.

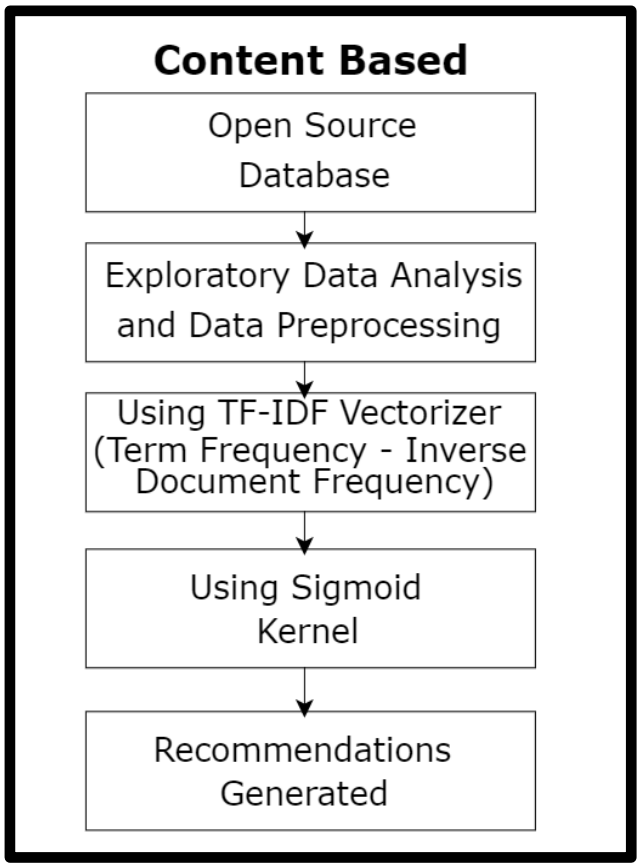

Figure 2: Workflow for Content Based Filtering

2.1.2 Additional Materials for Study 1:

The database used is Kaggle TMDB 5000 movie dataset.

2.2 Study 2: Collaborative Filtering using Matrix Factorization

Collaborative Filtering method works on similarity based on past interactions of various users with the targeted item (Seif, 2019). In this method, all the past user interaction information with the target item is gathered, and then it is converted into an interaction matrix, this matrix is then mined to decide what items can be recommended to the other users with similar taste (AltexSoft, 2021). It also helps to surprise

the user with new and diverse content which they may like. It can be divided into two parts:

Memory Based

Model Based

Memory Based Filtering: It solely believes and works on the memory of the past data provided by the user (AltexSoft, 2021). It simply uses the distance-measurement approach to form clusters of similar users or items (AltexSoft, 2021). It is simple and easy but unable to provide real-time recommendations (AltexSoft, 2021). This method can be achieved by K-Nearest Neighbors.

It can further be divided into two parts:

User-User: In this, user clusters are created using the similarity basis of interests, and then the items liked by one user are recommended to the others present in the cluster hoping they will like it.

Item-Item: In this method, clusters are formed on the basis of similarity between items. If an item is rated good by the customer, then the other similar items present in that cluster are recommended to them.

Model Based Filtering: This method is more reliable as it first assembles the data into a detailed model of user, items and ratings and then uses it to recommend items. It also helps in finding some underlying patterns which give far better results (AltexSoft, 2021). It provides real time recommendations. It can be achieved by using Matrix Factorization, Deep Neural Networks, etc.

We have used Matrix Factorization which is Model Based Collaborative Filtering to obtain the desired recommendations.

Matrix factorization works on the principle of reducing dimensionality of a bigger matrix into two or more smaller matrices (Kordík, 2019). For example: We have a matrix [a x b], we tend to divide it using k latent components such as: [a x k] and [k x b].

$$[a\ x\ b]\ \simeq\ [a\ x\ k].[k\ x\ b]$$

This approach tries to predict the user-item ratings by multiplying the corresponding rows and columns (Seif, 2019) and taking probable errors into account (AltexSoft, 2021). This helps to predict how a user will react to a certain item if it is recommended to them. In simpler words, it tells how likely a user will enjoy a certain product. There are various methods to achieve Matrix Factorization: Funk MF, SVD, Deep-Learning MF, PCA, etc. It is still one of the most used techniques for recommendation systems.

We have used SVD which stands for Singular Value Decomposition to obtain the desired recommendations.

SVD (Singular Value Decomposition) performs linear dimensionality reduction by means of truncated SVD. It can work with sparse matrices efficiently. The elements present in this matrix are the ratings that are given to the items by the users (Kumar, 2020).

The SVD of a [m x n] matrix A is given by:

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

where, U - mxn matrix of orthogonal eigenvectors of AA^T.

V^T - transpose of a nxn matrix containing the orthogonal eigenvectors of A^{T}A.

W - a nxn diagonal matrix of the singular values which are the square roots of the eigenvalues of A^TA (Saxena, 2022).

2.2.1 Design of Study 2:

In this, we have first analyzed and preprocessed the data. After this, we have applied Matrix Factorization by first creating a pivot table of user-item ratings and then transposing it. Now, SVD is used for linear dimensionality reduction. Following this, we have mined this matrix to calculate the correlation coefficient of each movie based on the user-item ratings. The higher the correlation coefficient, the more the user is likely to enjoy a certain movie. The dataset has user-item ratings information, which is used to generate the recommendations.

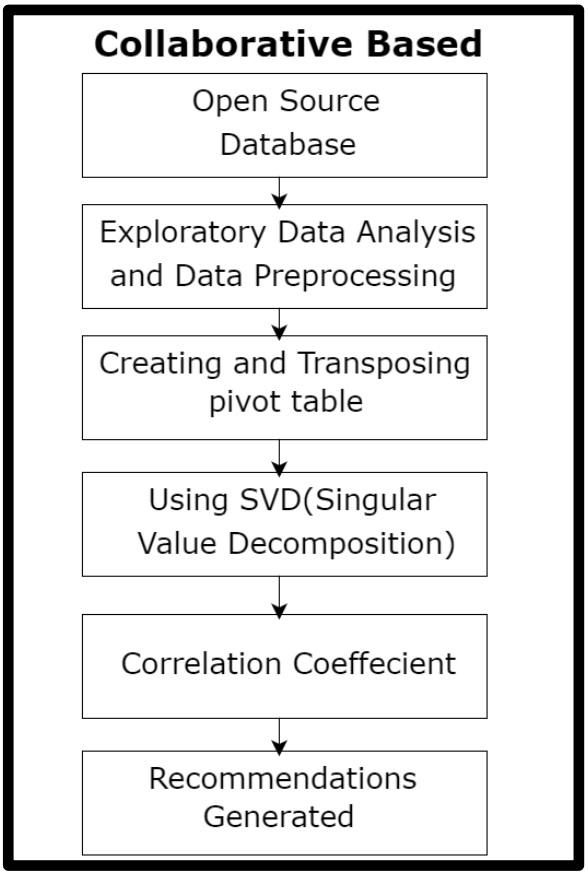

Figure 3: Workflow for Collaborative Filtering

2.2.2 Additional Materials for Study 2:

The database used is MovieLens dataset.

2.3 Study 3: Hybrid Filtering using Weighted Rating Score

Most of the systems nowadays use this approach to recommend generations. As the name suggests, this method tends to combine both content-based and collaborative approaches by limiting their disadvantages and increasing the advantages. Its goal is to ensemble more than one technique to provide more accurate recommendations. It is a collaborative filtering mixed with various techniques and

algorithms to reduce the cold start problem (when we don't have enough data on the user) (AltexSoft, 2021).

Weighted Rating Score is used to generate the desired recommendations.

Weighted Rating Score helps us to prioritize the items on various basis. It basically helps to derive a quantitative value for each item in our dataset. We can then use these values to prioritize the items (*Weighted Scoring | Definition and Overview*, n.d.).

Weighted Rating can be calculated by using this formula:

$$W = ((R.v) + (C.m))/(v + m)$$

where, W - Weighted Rating

v - number of votes for the item

m - minimum votes required to be listed in top

C - the mean vote across the whole report (currently 6.9) (*IMDb*, n.d.) (*Help*, n.d.)

2.3.1 Design of Study 3:

First of all, we have analyzed and preprocessed the data. After this, we have calculated the Weighted Rating Score using the formula mentioned above in heading 2.3. Then, we have simply used a "popularity" feature available in our database. It gives a popularity score based on the user ratings. Following this we analyzed the top movies generated on the basis of Weighted Rating Score and popularity separately by plotting a bar graph. Now, MinMax Scalar is used to normalize the values of popularity and weighted score. Then, we calculated a final score giving 50 percent priority to both popularity and weighted rating score. It can calculate using a basic formula which is:

$$Final\ Score = (Weighted\ Rating\ Score \times 0.5) + (Popularity \times 0.5)$$

The dataset has all the information required to calculate these scores. Features such as "vote_average", "vote_count" and "popularity".

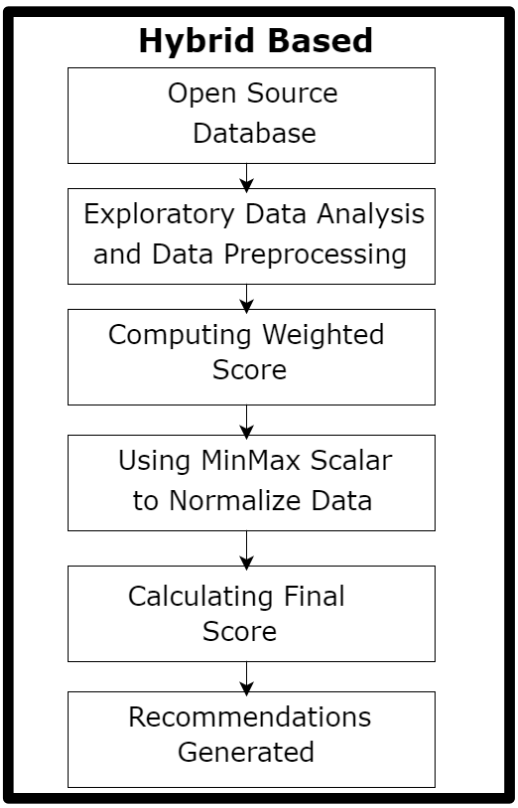

Figure 4: Workflow for Hybrid Filtering

2.3.2 Analysis Methods for Study 3:

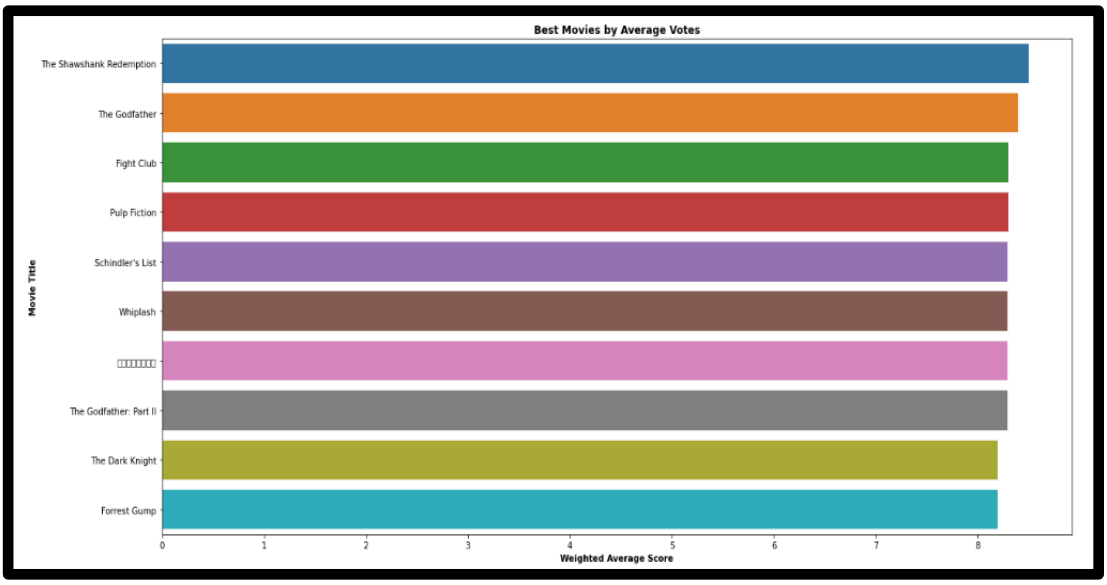

Figure 5: Top Movies based on Weighted Rating Score

We analyzed the top movies generated by Weighted Rating Score using a bar graph as shown in Figure 5. The X-axis represents the names of the movies and Y-axis shows the weighted average score. It can be done through either a bar graph or a very simple ".head()" function present in pandas library.

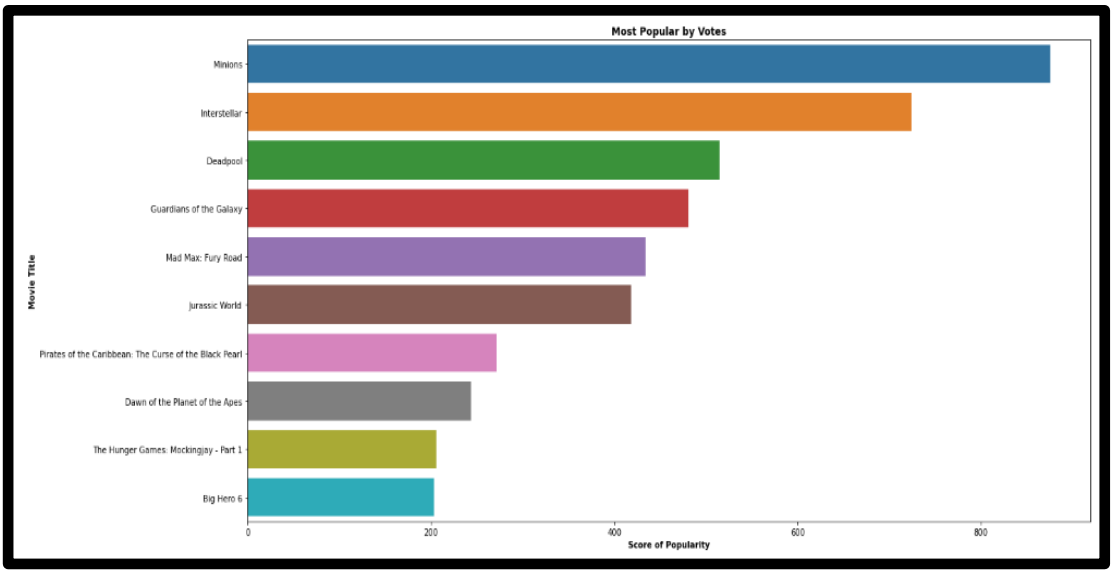

Figure 6: Top Movies based on "popularity" feature

As shown in Figure 6 these are the top movies generated from the "popularity" feature available in the database. The X-axis represents the names of the movies and Y-axis shows the popularity of that movie. It can be achieved by either a bar graph or a very easy ".head()" function provided by pandas library.

2.3.3 Additional Materials for Study 3:

The database used is Kaggle TMDB 5000 movie dataset.

3. Results

3.1 Results of Study 1

```
1  recommend("Avatar")
```

```
1341                  Obitaemyy Ostrov
634                         The Matrix
3604                         Apollo 18
2130                     The American
775                          Supernova
529                    Tears of the Sun
151                            Beowulf
311       The Adventures of Pluto Nash
847                            Semi-Pro
942                  The Book of Life
36      Transformers: Age of Extinction
570                              Ransom
1610                              Hanna
3070           Jeff, Who Lives at Home
1784                         EverAfter
1033                           Insomnia
2578                         The Marine
1013                          Child 44
3724                     Falcon Rising
4211           La sirène du Mississipi
```

Figure 7: Recommendations Generated from Content based Filtering

In Figure 7 the result of content-based filtering can be seen. These are the top 20 movies which are related to "Avatar" based solely on the "overview" feature which is the plot of the movie.

3.2 Result of Study 2

```
1  Recommend = Correlation[Wizard]
2  list(Movie_Title[(Recommend >= 0.9)])
```

```
['Animal House (1978)',
 'Beetlejuice (1988)',
 'Big (1988)',
 "Bug's Life, A (1998)",
 'Christmas Story, A (1983)',
 'Crocodile Dundee (1986)',
 'E.T. the Extra-Terrestrial (1982)',
 'Fantasia (1940)',
 'Goonies, The (1985)',
 'Grease (1978)',
 'Honey, I Shrunk the Kids (1989)',
 'Jaws (1975)',
 'Jungle Book, The (1967)',
 'Lady and the Tramp (1955)',
 'Little Mermaid, The (1989)',
 'Mary Poppins (1964)',
 'Romancing the Stone (1984)',
 'Sound of Music, The (1965)',
 'Splash (1984)',
 'To Kill a Mockingbird (1962)',
 'Who Framed Roger Rabbit? (1988)',
 'Wizard of Oz, The (1939)']
```

Figure 8: Recommendations Generated from Collaborative Filtering

The movies having higher correlation scores than 0.9 are shown in Figure 6. These are the top movies based on user-item interaction ratings.

3.3 Result of Study 3

```
1  Recommend.head(10)
```

```
95                                        Interstellar
546                                            Minions
94                                Guardians of the Galaxy
788                                           Deadpool
127                                Mad Max: Fury Road
3865                                          Whiplash
65                                      The Dark Knight
1881                              The Shawshank Redemption
3337                                     The Godfather
199     Pirates of the Caribbean: The Curse of the Bla...
```

Figure 9: Recommendations Generated from Hybrid Filtering

In Figure 7 the top 10 movies generated from Hybrid Filtering can be seen. The recommendations are generated from Weighted Rating Score and the "popularity" feature available in the database.

4. Discussion and Synthesis

The recommendations generated from all the three methods: Content Based Filtering, Collaborative Filtering and Hybrid Filtering have been shown in Figure 5, Figure 6, Figure 7.

In this paper, Content Based Filtering is used to generate the recommendations based on similarity between movies. If a user watches a particular movie, then other movies similar to it will be shown as recommendations to them.

Collaborative Filtering is implemented to generate real-time recommendations using user-item rating interaction matrix to target specific users. Here, I have targeted every user and not a particular group of users.

Hybrid Filtering is done to generate the most popular movies of all time based on Weighted Rating Score which I calculated using votes given to the movie by users and popularity of that particular movie. Fifty percent priority is given to both of them so that recommendations generated could be much more accurate and meaningful.

4.1 Future Scopes

In this project, I have used 1st and 2nd generation techniques, but there is so much more that has been contributed and that can be done in the field of recommendation systems.

We can add 3rd generation techniques such as Deep Neural Network, Natural Language Processing, and so on.

Natural Language Processing techniques can be used to resolve the cold start issue in a much more effective way.

Hidden Markov Model methods can be applied to detect the change in user preference.

Various algorithms and techniques can be combined together to make an effective recommendation system.

5. Conclusions

Currently we are living in a world where time is very crucial and there is a vast variety of products and items available to go through. A user cannot go through each and every item. To reduce their struggle and time, recommendation systems are really necessary. In addition to this, they are really driving the economy in a much more efficient way. “Netflix claims that 80% of watched content is based on algorithmic recommendations” (Chhabra, 2017). From choosing books to recommending movies or e-products, it is used everywhere. Every system serves its own purpose and generates personalized recommendations which are very user-oriented and user-friendly.

Content Based Filtering is user independent, so it does not encounter data sparsity problems. It shows transparency about how recommendations are generated. It has a shortcoming which is the user gets limited to content which is alike with the previous content they have seen.

On the other hand, Collaborative Filtering is greatly successful in surprising users. It identifies the underlying pattern in changes in user preferences. Moreover, it is flexible and tries to identify the character

of the user and shows diverse content which the user may like. Unlike Content Based Filtering, it faces the cold start issue and data sparsity problems.

The most used and most triumphant method is Hybrid Filtering, as it ensembles both Content Based and Collaborative Filtering in such a way that it tends to reduce their weaknesses and promote their strengths.

Acknowledgement: I would like to express my deepest gratitude to my Professor Dr. Sorin Istrail for his infinite motivation, attention, support, patience, and immense knowledge throughout the project. I would like to thank Chen Wang for continuously helping me to upgrade my work. I could not have imagined having a better mentor and advisor for my research-based project.

Appendix A: Code for Content Based Filtering

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import sigmoid_kernel
movie=pd.read_csv('tmdb_5000_movies.csv')
credit=pd.read_csv('tmdb_5000_credits.csv')
movie.head()
credit.head()
credit['id']=credit['movie_id']
credit.drop('movie_id', axis=1, inplace=True)
data=movie.merge(credit, on='id')
data.head()
data.drop(['homepage', 'title_x', 'title_y', 'status', 'production_countries'], axis=1, inplace=True)
data['overview']
TF=TfidfVectorizer(stop_words='english', analyzer='word', min_df=3, strip_accents='unicode', ngram_range=(1,3), token_pattern=r'\w{1,}', max_features=None)
vec=TF.fit_transform(data['overview'])
vec
sig=sigmoid_kernel(vec,vec)
sig
index=pd.Series(data.index, index=data['original_title'])
index
index['Shanghai Calling']
sig[4801]
list(enumerate(sig[index['Shanghai Calling']]))
def recommend(title, sig=sig):
    place=index[title]
    sig_score=list(enumerate(sig[place]))
    sig_score=sorted(sig_score, key=lambda x:x[1], reverse=True)
    sig_score=sig_score[1:21]
```

```
    indices=[i[0] for i in sig_score]
    return data['original_title'].iloc[indices]
recommend("Avatar")
```

Appendix B: Code for Collaborative Filtering

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from sklearn.decomposition import TruncatedSVD
import warnings
movie = pd.read_csv('movies.csv')
rating = pd.read_csv('ratings.csv')
movie.head()
rating.head()
movie.drop('genres', axis=1, inplace=True)
rating.drop('timestamp', axis=1, inplace=True)
data = movie.merge(rating, on='movieId')
data.head()
New_Data = data.dropna(axis=0, subset=['title'])
Rating_Count = (New_Data.groupby('title')['rating'].count().reset_index())
Rating_Count.head()
Rating_Count['Total_Rating_Count']=Rating_Count['rating']
Rating_Count.drop('rating', axis=1, inplace=True)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
print(Rating_Count['Total_Rating_Count'].describe())
print(Rating_Count['Total_Rating_Count'].quantile(np.arange(.9,1,.01)))
Final = New_Data.merge(Rating_Count, left_on='title', right_on='title', how='left')
Final.head()
Threshold = 50
New_Final = Final.query('Total_Rating_Count >= @Threshold')
```

```
New_Final.head()
Features = New_Final.pivot_table(values='rating', index='userId', columns='title').fillna(0)
Features.head()
Features.shape
X = Features.values.T
X.shape
SVD = TruncatedSVD(n_components=12, random_state=17)
Matrix = SVD.fit_transform(X)
Matrix.shape
warnings.filterwarnings("ignore", category=RuntimeWarning)
Correlation = np.corrcoef(Matrix)
Correlation.shape
Movie_Title = Features.columns
Movie_List = list(Movie_Title)
Wizard = Movie_List.index("Wizard of Oz, The (1939)")
print(Wizard)
Recommend = Correlation[Wizard]
list(Movie_Title[(Recommend >= 0.9)])
```

Appendix C: Code for Hybrid Filtering

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
movie=pd.read_csv('tmdb_5000_movies.csv')
credit=pd.read_csv('tmdb_5000_credits.csv')
movie.head()
credit.head()
credit['id']=credit['movie_id']
credit.drop('movie_id', axis=1, inplace=True)
data=movie.merge(credit, on='id')
data.head()
data.info()
data=data.drop(columns=['budget','genres','homepage','keywords','original_language','overview','production_companies','release_date','revenue','cast','crew','runtime','spoken_languages','status','tagline','title_x','title_y','production_countries'])
data.head()
R=data['vote_average']
v=data['vote_count']
C=data['vote_average'].mean()
m=data['vote_average'].quantile(0.70)
W=((R*v)+ (C*m))/(v+m)
data['Weighted_Score']=W
data.head()
Score=data.copy().sort_values('Weighted_Score', ascending=False)
Score.head()
plt.figure(figsize=(20,10))
sns.barplot(x=Score['Weighted_Score'][:10], y=Score['original_title'][:10], data=Score)
```

```
plt.title('Best Movies by Average Votes', weight='bold')
plt.xlabel('Weighted Average Score', weight='bold')
plt.ylabel('Movie Title', weight='bold')
plt.show()
popularity=data.sort_values('popularity', ascending=False)
popularity.head()
plt.figure(figsize=(20,10))
sns.barplot(x=popularity['popularity'][:10], y=popularity['original_title'][:10], data=popularity)
plt.title('Most Popular by Votes', weight='bold')
plt.xlabel('Score of Popularity', weight='bold')
plt.ylabel('Movie Title', weight='bold')
Scale=MinMaxScaler()
Scaled_Data=pd.DataFrame()
Scaled_Data['Weighted_Score']=data['Weighted_Score']
Scaled_Data['popularity']=data['popularity']
Scaled_Data=Scale.fit_transform(Scaled_Data)
Scaled_Data=pd.DataFrame(data=Scaled_Data, columns=['Weighted_Score','popularity'])
Scaled_Data.head()
data['Normalized_Score']=Scaled_Data['Weighted_Score']
data['Normalized_Popularity']=Scaled_Data['popularity']
data.head()
data.drop(['popularity','vote_count','vote_average','Weighted_Score'], axis=1, inplace=True)
data.head()
data['Final_Score']=data['Normalized_Score']*0.5+data['Normalized_Popularity']*0.5
data.head()
Final=data.sort_values(['Final_Score'], ascending=False)
Final.head(20)
Recommend = Final['original_title']
Recommend.head(10)
```

References:

AltexSoft. (2021, July 27). "Recommender Systems: Behind the Scenes of Machine Learning-Based Personalization. AltexSoft. Retrieved July 06, 2022, from https://www.altexsoft.com/blog/recommender-system-personalization/.

Kordík, P. (2019, December 15). Machine Learning for Recommender Systems - Part 1 (Algorithms, Evaluation and Cold Start). Medium. Retrieved July 07, 2022, from https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed.

Seif, G. (2019, September 4). An Easy Introduction to Machine Learning Recommender Systems. KDnuggets. Retrieved July 07, 2022, from https://www.kdnuggets.com/2019/09/machine-learning-recommender-systems.html

Wikimedia Foundation. (2022, July 18). Recommender System. Wikipedia. Retrieved July 07, 2022, from https://en.wikipedia.org/wiki/Recommender_system.

Saxena, P. (2022, March 14). Singular Value Decomposition (SVD). GeeksforGeeks. Retrieved July 14, 2022, from https://www.geeksforgeeks.org/singular-value-decomposition-svd/

Hui, J. (2022, February 08). Machine Learning — Singular Value Decomposition (SVD) & Principal Component Analysis (PCA). Medium. Retrieved July 14, 2022, from https://jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491

Scott, W. (2019, February 15). TF-IDF for Document Ranking from scratch in python on real world dataset. Towards Data Science. Retrieved July 19, 2022, from https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Simha, A. (2021, October 6). Understanding TF-IDF for Machine Learning. Capital One. Retrieved July 19, 2022, from https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

Dilmegani, C. (2017, August 11). Recommendation Systems: Applications and Examples in 2022. AIMultiple. Retrieved July 15, 2022, from https://research.aimultiple.com/recommendation-system/

Kumar, V. (2020, March 25). Singular Value Decomposition (SVD) In Recommender System. Analytics India Magazine. Retrieved July 26, 2022, from https://analyticsindiamag.com/singular-value-decomposition-svd-application-recommender-system/

Weighted Scoring | Definition and Overview. (n.d.). ProductPlan. Retrieved July 16, 2022, from https://www.productplan.com/glossary/weighted-scoring/

IMDb. (n.d.). Wikipedia. Retrieved July 16, 2022, from https://en.wikipedia.org/wiki/IMDb

Help. (n.d.). IMDb | Help. Retrieved July 16, 2022, from https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#

Chhabra, S. (2017, August 22). Netflix says 80 percent of watched content is based on algorithmic recommendations. MobileSyrup. Retrieved July 29, 2022, from https://mobilesyrup.com/2017/08/22/80-percent-netflix-shows-discovered-recommendation/

Gupta, U., & Patil, N. (2015, July 13). Recommender system based on Hierarchical Clustering algorithm Chameleon. 2015 IEEE International Advance Computing Conference (IACC).

Kumar, M., Yadav, D.K., Singh, A., & Gupta, V. K. (2015). A Movie Recommender System: MOVREC. International Journal of Computer Applications(0975 – 8887), 124(3).

Kużelewska, U. (2014). Clustering Algorithms in Hybrid Recommender System on MovieLens Data",. Studies in Logic, Grammar and Rhetoric.

Campos, L. M. d., Fernández-Luna, J. M., Huete, J. F., & Rueda Morales, M. A. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. International Journal of Approximate Reasoning.

Chiru, C.-G., Dinu, V.-N., Preda, C., & Macri, M. (2015). Movie Recommender System Using the User's Psychological Profile. IEEE International Conference on ICCP.

Lin, H., Yang, X., & Wang, W. (2014, August 27). A Content-Boosted Collaborative Filtering Algorithm for Personalized Training in Interpretation of Radiological Imaging J Digit Imaging.