ISSN No. 0976-5697

Volume 11, Special Issue I, May 2020

International Journal of Advanced Research in Computer Science

CONFERENCE PAPER

Available Online at www.ijarcs.info

MOVIE RECOMMENDATION SYSTEM

Prajwal S
School of C & IT
REVA University,Bengaluru, India
meisprajwal@gmail.com

Sharan Kumar P
School of C & IT
REVA University,Bengaluru, India
sharankmr31@gmail.com

Sagar Chavan
School of C & IT
REVA University,Bengaluru, India
sagarsc01@gmail.com

Prof. Anil Kumar Ambore
School of C & IT
REVA University,Bengaluru, India
anilambore@reva.edu.in

Sharath Kumar T L
School of C & IT
REVA University,Bengaluru, India
sharathg138@gmail.com

Abstract—In this digital era that we live in, Recommendation systems have become a part and parcel of our everyday lives. There are tons of options out there for everything that we do and people might find themselves in a difficult place while making a choice, 'a perfect one!' This is where the Recommendation Systems step in. Internet Giants such as Amazon, Netflix, YouTube, Spotify, Facebook etc can be seen using these technologies to keep their audience interested. So, through this work, we attempt to build a simple movie recommendation system employing the famous technique'User-User Collaborative Filtering' and design a GUI for the same.

Keywords—recommender system, collaborative filtering, user based collaborative filtering, cosine similarity.

I. INTRODUCTION

Generally, Recommendation Systems are algorithms intended to suggest relevant items to users (items being videos or movies to watch, music to listen, products to buy or anything else depending on enterprises). Recommender Systems have a vast category of applications . We can observe a dramatic increase in the prominence of these technologies particularly on the online channels . The products that are recommended on these online channels range from a song or a film on web streaming apps to people on dating websites and career forums or even something naive like a web search on the search engines. In most cases, these systems are so advanced that they are capable of collecting information about users' tendencies resulting in the knowledge that can be used in the future to enhance their recommendations. YouTube, for example, will track your viewing history on their website and decide what kinds of videos are appealing to you. Such systems also often improve their suggestions based on the activities that are similar between a group of people. For example, if a shopping website recognises that a large number of customers who purchased a professional camera have purchased a carrying case for it, then the algorithm may recommend the carrying case to the new consumer who is viewing a camera on their site.

Users often expect outstanding recommendations because of the developments in Recommender Systems. They have a small tolerance for programs which can't offer appropriate suggestions. For example, if a music streaming app can't predict and play the music the user likes, then the user will eventually get bored and the app wouldn't seem great to him. This has brought the Recommender Systems to the spotlight and the tech companies focusing on improvising their models. But, the problem here is not as simple as it may appear. Each user has his/her owntendencies. Adding to this, the problem gets even more complex as the user's tendencies are also influenced largely by factors, such as the time of the day, his surroundings, his mood etc. For example, a user might like to listen to a higher tempo genre such as rock while he's exercising and maybe a low-tempo genre such as smooth-jazz.

Collaborative filtering (CF) algorithm is the most common technique employed in Recommender Systems. It is widely used due to its personalised recommendation. The main idea behind this technique is to utilize neighbours who have similar likings as of the active user and suggest the recommendations based on their liking history [6]. The CF algorithm is essentially built on the following three postulates: Users share comparable choices and likings; Their selections and likings are consistent; Their likings can be predicted based on their selections. Owing to the above postulates, comparison of users' behaviours and their tendenciesis the base of the CF algorithm. In this work, we will attempt to build a simple Recommendation System employing this widely popular technique.

The primary step of the CF algorithm is to retrieve the user's watch history, which can be interpreted as a matrix of ratings, with the user's score given to a particular movie as an entry[6]. The matrix is a table with rows and columns representing individual users and unique movies. Thus, the rating value of the user is denoted by the numerical value at the intersection of a row and column. A missing rating score at this intersection means that the item has not yet been rated by the particular user. Because of this sparse scoring problem, we have to ensure that the matrix is complete without any missing values to maintain our model's accuracy and consistency.

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}} \qquad (1)$$

The next step is to measure the users similarity and identify the nearest neighbours to the active user. There are several techniques for assessing similarities. We have used the Cosine similarity measure method in our case, it's calculated using the following equation:

In the above equation 'rx' is the user rating of 'x' on item 's' and 'ry' is user y rating on item 's', 'Sxy' means the items co-evaluated by the user 'x' and 'y'.

The final step is to determine the rating of objects. The rating is based on the weighted average of ratings by the neighbour users.[1].

$$r_{x,s} = \overline{r}_x + \frac{\sum_{y \in S_{xy}} (r_{y,s} - \overline{r}_x) sim(x,y)}{\sum_{y \in S_{xy}} sim(x,y)} \qquad (2)$$

$\overline{r}_x$ is the average rating of user x.

Explanation above indicates that the CF algorithm consumes high computational time and compute resources. The computation cycle will continue for several hours or even longer, if the data set is very large. Therefore, we suggest a system in which the scoring function is restricted only to movies considering the active user's nearest 'K' neighbours, which is derived from the rule 'Neighbourhood for User (K).

II. EXISTING WORKS

Researchers have developed various film recommendation strategies to recommend films to the consumer according to their desires or preferences. Because recommendation systems are such a hot topic in recent data science research, several scientific papers on recommendation systems have been published. Finally, several researchers presented genuine article relating to our project.[8]

Existing Methods for movie recommendation:

A. A paper was published in 1998 with focus on Recommender Systems. It was the first paper in this topic. A fair number of articles have been published since. Different reasons for increasing the efficiency of the recommender program have been clarified. John O 'Donovar n, Barry Smyth, took confidence in the year 2005 as the percentage of accurate predictions that a person has made in general (person-level trust) or in respect of a particular item (item-level trust)[1]. The key goal of our project is to recommend movies to users according to their preferences. The system should be able to look at the data on videos in the database and provide these videos to people who might like them. Recommendation Systems are used to provide automatic recommendations to users of a service by using user's behaviors from the past. There are a lot of algorithms available for recommender systems. Thus, choosing one among all of these is a difficult task. This decreases the prediction error by 22%.

B. Lops et al. (2011) [2], presented item based collaborative filtering recommendation algorithm which resolves the problem arrived in rating. Rating of user to user based collaborative filtering by using the rating distribution per item, not for user. This leads to more stable rating distribution in the model, so the model doesn't have to rebuild as often. By using this algorithm accuracy of 75% was achieved by the system.

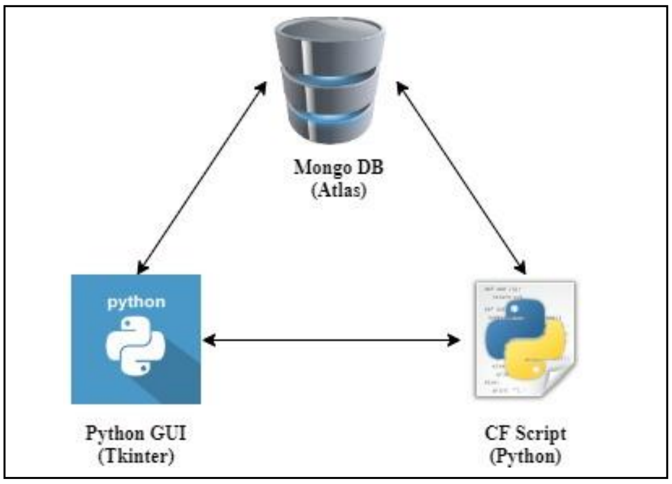
C. Collaborative filtering and content driven filtering have been classified into recommendation systems. This technique was known to highlight two major problems: problem of sparsity and problem of scalability. Burkey (2007)[3] proposed a hybrid device capable of solving this problem and achieving 70 percent accuracy.

D. D. Hongli Lin et al.(2008)[4] introduced a process called Content-Boosted Collaborative Filtering (CBCF). The algorithm is divided into two stages: First, content-based filtering, which enhances existing trainee case ratings and second, collaborative filtering, which provides the final predictions. The CBCF algorithm contains all the benefits of CBF and CF, thus also addressing all of their disadvantages. There are various types of recommender systems with different methods, some of which are listed as below: Content-based Filtering Systems (CBF-based systems), as this method is 75 percent accurate.

E. Eugene Seo and Ho-Jin Choi (2009) [5], presented the k-means Clustering method is superior to other algorithm and more accurate to predict the rating and review. It gives great sense of accuracy of about 75%.

F. Costin-Gabriel Chiru et al.[6], suggested Movie Recommender, a program that uses user-known knowledge to make recommendations for films. This program attempts to solve the problem of unique suggestions resulting from ignoring the user-specific data. The user's psychological profile, their history of viewing and the data from other websites that contain movie scores are collected. We are founded upon the estimation of aggregate similarity. This program has introduced a hybrid algorithm and is 79 per cent effective.

III. PROPOSED SYSTEM



We have proposed a simple, end to end system which comprises a basic GUI and connected to an online instance of

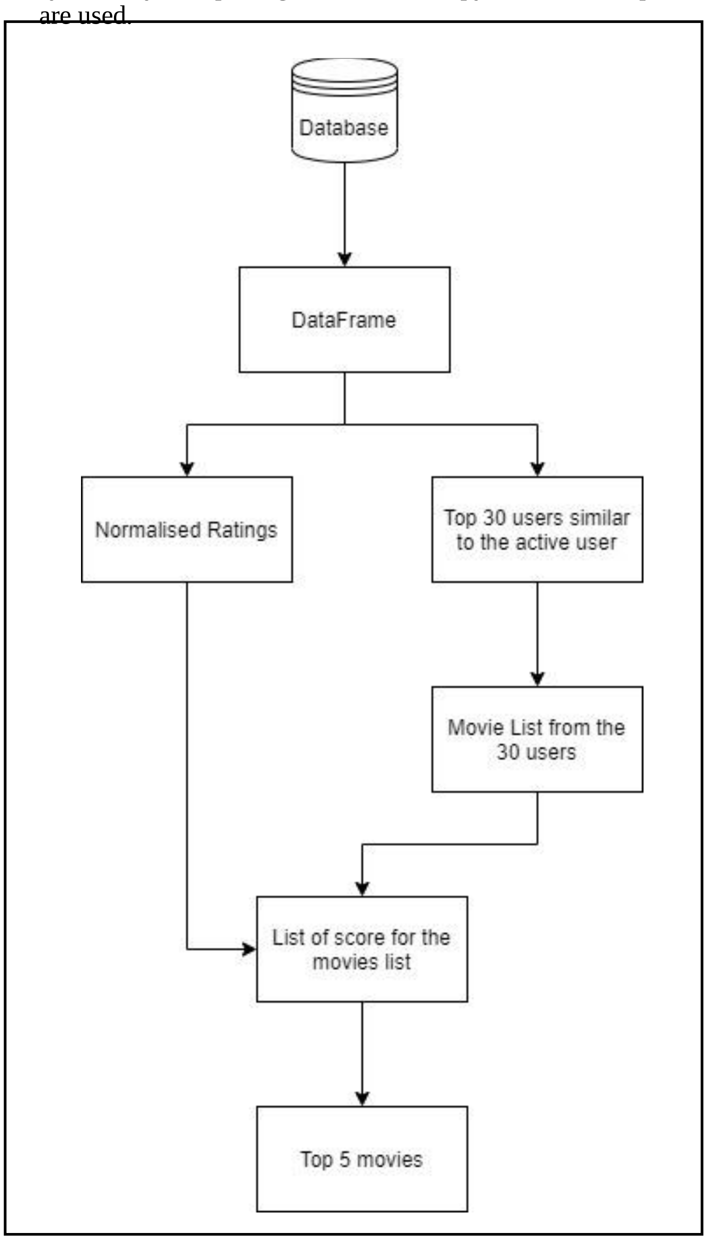
MongoDB. The system can be broken down to three major components, namely:

A. Python GUI (Tkinter)

'Tkinter', is the standard GUI bundled with Python. It is readily available with Python installs on Linux, Windows and MacOS. We have used this tool to build a simple GUI for our Recommendation System with a Login/Register Page and a User Page where Recommendations are listed.

B. CF Script (Python)

The CF Script is the main driver code of the system. It is written using Python 3.7 and contains the User-User Collaborative Filtering function which is the base of our system. Python packages such as numpy, sklearn and pandas are used.



Database
DataFrame
Normalised Ratings
Top 30 users similar to the active user
Movie List from the 30 users
List of score for the movies list
Top 5 movies

Fig 3.2Model Flowchart

C. Mongo DB (Database)

An online instance of a MongoDB, a non-relational database, has been implemented on the Atlas platform. This stores the user login information and the datasets our system is dependent on.

A user can run the program and login using his credentials into the system on the GUI or create a new account if he is new. The login details are verified with the MongoDB hosted online. After a successful login, the user is taken into his/her personal page where he can find his recommendations or edit his profile details. The CF script is triggered with the user's id to display the recommendations on the GUI.

IV. Methodology

A. Score Function

The score function generates a score that evaluates how strong the user 'u' likes / prefers an object 'i'. Typically, this is achieved by considering other users' ratings who share similarities with the active user. The formula employed here is:

$$s(u,i) = \bar{r}_u + \frac{\sum_{v\in V}(r_{vi} - \bar{r}_v) * w_{uv}}{\sum_{v\in V} w_{uv}} \quad (3)$$

In this equation,'s' is the predicted score for the item, 'u' is the active user, 'i' is the item in consideration, 'r' is the user's rating and 'w' is the weight. The score is determined as the sum of the ratings that each user gave that particular item and subtracting that user's average rating. This sum is then compounded by a weight which expresses a users' similarity or in other words how largely the user is expected to contribute to other user's predictions. Thescore can vary from 0 to 1 where 0 is the minimum and 1 is the maximum.

B. Cosine Similarity

To measure the weight in the formula (mentioned above) which is used to determine how close the users are, Cosine Similarity is used. It is based on the scores which have been rated by both users in the past. Once matrix of user ratings is normalized, this function is enforced it to find similarities. The function is borrowed from sklearn library in Python.

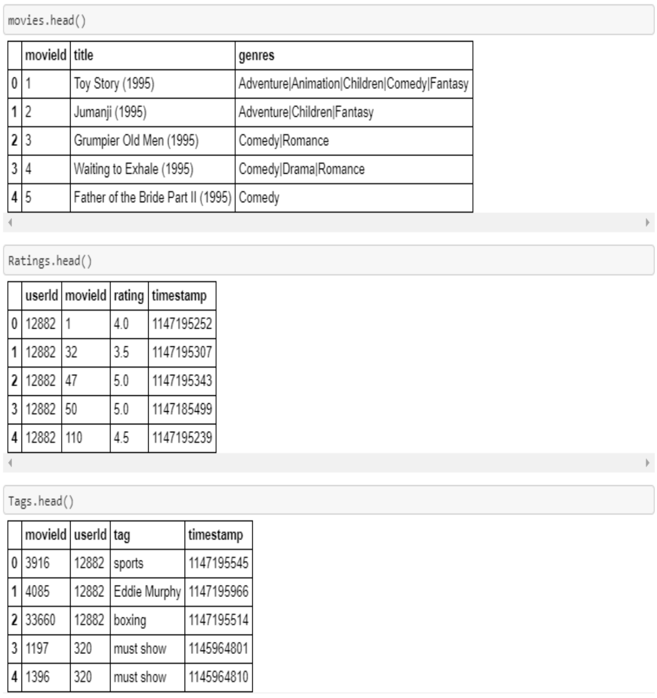
C. Neighbourhood for User (K)

This function is mainly to address the complexity of the problem. Recommendation Systems work on large datasets. Even a simple analysis task would take a significantly long time. Therefore, it is necessary to emphasize preserving and collecting only the appropriate and necessary highlights from the dataset to solve the complexity issue.

In our case the matrix obtained for calculating similarity is (862*862), as there are 862 unique users in the dataset. Therefore, we create a notion of neighbourhood to resolve this complexity issue. This only includes the set of (K) identical users who share similarities with a given user. We took the value of k as 30, in our test run. So, we would have 30 nearest neighbours for all the users to consider the movies that they have rated as consideration for the active user. This function takes the matrix of similarity and the value of n as the input and returns the users' nearest 'n' neighbours.

D. User-User Collaborative Filtering

Step 1: Import the data from the Movie Lens Dataset namely 'Movies', 'Ratings' and 'Tags' as separate Data Frames (Pandas).



```
movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
Ratings.head()
```

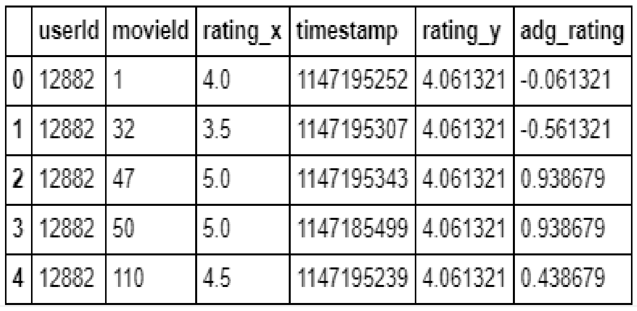
	userId	movieId	rating	timestamp
0	12882	1	4.0	1147195252
1	12882	32	3.5	1147195307
2	12882	47	5.0	1147195343
3	12882	50	5.0	1147185499
4	12882	110	4.5	1147195239

```
Tags.head()
```

	movieId	userId	tag	timestamp
0	3916	12882	sports	1147195545
1	4085	12882	Eddie Murphy	1147195966
2	33660	12882	boxing	1147195514
3	1197	320	must show	1145964801
4	1396	320	must show	1145964810

Fig 4.1 DataFrames

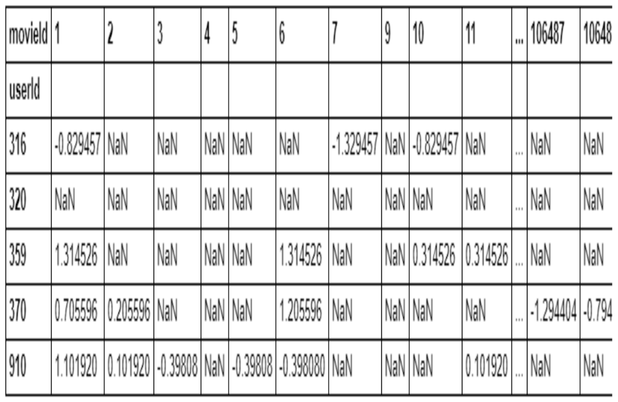
Step 2: Normalise the user ratings.



	userId	movieId	rating_x	timestamp	rating_y	adg_rating
0	12882	1	4.0	1147195252	4.061321	-0.061321
1	12882	32	3.5	1147195307	4.061321	-0.561321
2	12882	47	5.0	1147195343	4.061321	0.938679
3	12882	50	5.0	1147185499	4.061321	0.938679
4	12882	110	4.5	1147195239	4.061321	0.438679

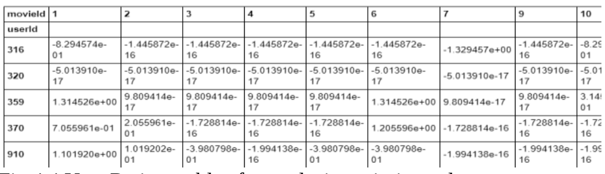
Fig 4.2 Normalised Ratings

Step 3: Fill the missing ratings in the user ratings table by taking the average of the movie ratings.



movieId	1	2	3	4	5	6	7	9	10	11	...	106487	10648
userId													
316	-0.829457	NaN	NaN	NaN	NaN	NaN	-1.329457	NaN	-0.829457	NaN	...	NaN	NaN
320	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
359	1.314526	NaN	NaN	NaN	NaN	1.314526	NaN	NaN	0.314526	0.314526	...	NaN	NaN
370	0.705596	0.205596	NaN	NaN	NaN	1.205596	NaN	NaN	NaN	NaN	...	-1.294404	-0.794
910	1.101920	0.101920	-0.39808	NaN	-0.39808	-0.398080	NaN	NaN	NaN	0.101920	...	NaN	NaN

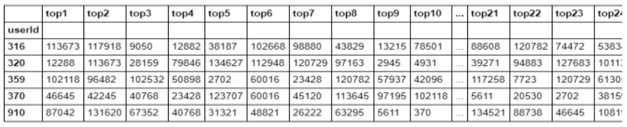
Fig 4.3 User Ratings table with missing values



movieId	1	2	3	4	5	6	7	9	10
userId									
316	-8.294574e-01	-1.445872e-16	-1.445872e-16	-1.445872e-16	-1.445872e-16	-1.445872e-16	-1.329457e+00	-1.445872e-16	-8.2 01
320	-5.013910e-17	-5.013910e-17	-5.013910e-17	-5.013910e-17	-5.013910e-17	-5.013910e-17	-5.013910e-17	-5.013910e-17	-5.01 17
359	1.314526e+00	9.809414e-17	9.809414e-17	9.809414e-17	9.809414e-17	1.314526e+00	9.809414e-17	9.809414e-17	3.14 01
370	7.055961e-01	2.055961e-01	-1.728814e-16	-1.728814e-16	-1.728814e-16	1.205596e+00	-1.728814e-16	-1.728814e-16	-1.72 16
910	1.101920e+00	1.019202e-01	-3.980798e-01	-1.994138e-16	-3.980798e-01	-3.980798e-01	-1.994138e-16	-1.994138e-16	-1.99 16

Fig 4.4 User Ratings table after replacing missing values

Step 4: Find the similarity of the active user with other users and find N (30 in our case) nearest neighbours.

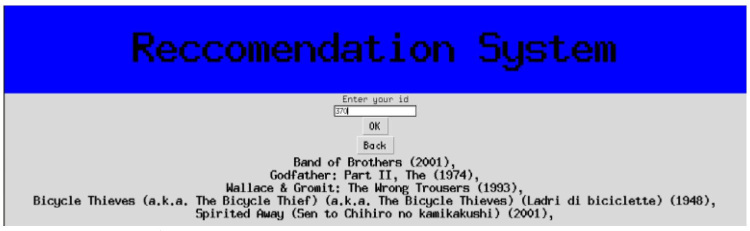


	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10	...	top21	top22	top23	top2
userId															
316	113673	117918	9050	12882	38187	102668	98880	43829	13215	78501	...	88608	120782	74472	5383
320	12288	113673	28159	79846	134627	112948	120729	97163	2945	4931	...	39271	94883	127683	1011
359	102118	96482	102532	50898	2702	60016	23428	120782	57937	42096	...	117258	7723	120729	6130
370	46645	42245	40768	23428	123707	60016	45120	113645	97195	102118	...	5611	20530	2702	3815
910	87042	131620	67352	40768	31321	48821	26222	63295	5611	370	...	134521	88738	46645	1081

Fig 4.5 User Similarity Matrix

Step 5: For the list of all the movies that the N neighbours have seen except the movies that the active user has already watched, generate a score using the Score Function.

Step 6: With the scores list arranged in descending order, pick the top 5 movies in the list and suggest them to the users.



Band of Brothers (2001),
Godfather: Part II, The (1974),
Wallace & Gromit: The Wrong Trousers (1993),
Bicycle Thieves (a.k.a. The Bicycle Thief) (a.k.a. The Bicycle Thieves) (Ladri di biciclette) (1948),
Spirited Away (Sen to Chihiro no kamikakushi) (2001),

Fig 4.6Snapshot of the GUI displaying the recommendations

V. RESULT

In this section, we show the end-results of our system. We successfully implemented a User-User Collaborative Filtering Technique in a Recommendation System. The system was tested on a Laptop running Ubuntu 16.04 LTS with an Intel i5 6200u processor and 8gb ram and 256 gb Ram with a WLAN connection to the internet. Our testing wasdone on the MovieLens data set with 862 unique users and 2501 movies. Our first run gave us a recommendation with a score(u,i) = 4.25576643.

VI. CONCLUSION

In this Paper, we successfully built a Recommendation System with User-User Collaborative Filtering Technique. The system worked seamlessly but it faced a major issue in the form of cold-start. The accuracy and consistency is highly dependent on the pre-existing knowledge about the user's preferences and likings. So, a new user without any pre-recorded information about him would not receive good recommendations when compared to a user with already existing information about his likings and preferences.

In our future works, we plan to try different approaches improve the model accuracy and to overcome the data sparsity in the ratings matrix using techniques such as matrix factorisation etc and compare the performances of the same. We also would like to bring other aspects like genre, cast and language as the factors of our recommendations.

ACKNOWLEDGMENT

This project was developed under the guidance of Prof. Anil Kumar Ambore and with a tremendous support from the School of C & IT at REVA University.

REFERENCES

[1] Adomavicius G., Tuzhilin A., Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. on Knowledge and Data Engineering, 2005, 17(6): 734-749

[2] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 22(1):5-53.

[3] Baptise Rocca : “Introduction to Recommender Systems.” https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

[4] Abdur Rrahmaan Janhangeer : “Python GUI – Tkinter”. https://www.geeksforgeeks.org/python-gui-tkinter/

[5] T. K. Quan, I. Fuyuki and H. Shinichi, "Improving Accuracy of Recommender System by Clustering Items Based on Stability of User Similarity," 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06), Sydney, NSW, 2006, pp. 61-61.

[6] N. Mustafa, A. O. Ibrahim, A. Ahmed and A. Abdullah, "Collaborative filtering: Techniques and applications," 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), Khartoum, 2017, pp. 1-6..

[7] R. Zhang, Q. Liu, Chun-Gui, J. Wei and Huiyi-Ma, "Collaborative Filtering for Recommender Systems," 2014 Second International Conference on Advanced Cloud and Big Data, Huangshan, 2014, pp. 301-308.

[8] https://aihubprojects.com/movie-recommendation-system-ai-projects/

[9] https://docs.mongodb.com/manual/