International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 6 305 – 310

Hybrid Recommendation System Using Clustering and Collaborative Filtering

Roshni Padate
Assistant Professor
Computer Engineering Department
Fr. Conceicao Rodrigues College of Engineering, Bandra
Mumbai, India
roshni@frcrce.ac.in

Priyanka Bane
B.E. Student
Computer Engineering Department
Fr. Conceicao Rodrigues College of Engineering, Bandra
Mumbai, India
priyankabane56@gmail.com

Jayesh Kudase
B.E. Student
Computer Engineering Department
Fr. Conceicao Rodrigues College of Engineering, Bandra
Mumbai, India
jkudase@gmail.com

Adarsh Gupta
B.E. Student
Computer Engineering Department
Fr. Conceicao Rodrigues College of Engineering, Bandra
Mumbai, India
adarshgupta139@gmail.com

***Abstract*—** A recommendation system is a way of suggesting users a subset of possible choice from a set of choices. An example of recommendation system in e-commerce web applications is to recommend customers what they might like to purchase from a wide variety of items based on their recent purchase behavior or search history. Recommender systems are redefining their meaning in today's business oriented world by providing organizations an effective means of driving revenue. There are a lot of recommendation engines present at the moment. Some of them work solely on the user's perspective whereas some work in generalized manner. There is a need that these recommender system start providing users with out of the box choices. They should not be limited to one's individuality, rather other user preferences based on similarity should also be considered. Hence recommender systems play a major role in e-commerce domains and helps the business to achieve data intelligence. The proposed system explains various methods by which the user can be provided recommendations.

Keywords*-Recommender systems(RS) , Collaborative Filtering(CF), Content based filtering, Hybrid filtering, The Movie Database(TMDb),The Open Movie Database(OMDb)*

*****

I. INTRODUCTION

Recommender systems(RSs) is a tool that aims at providing suggestions to the users in order to help them in their decision making process such as which movies to watch, what type of music to listen. RSs have become a popular technique to prune large information spaces so that users are directed toward those items that best meet their needs and preferences. The architecture of RSs and their evaluation on real world problems is an active area of research. Thus RSs arose from practical requirements as personalized e-services are required in many application domains. The interest in RSs has dramatically increased because E-commerce is in its infancy in developing countries and hence they play an important role in this area. The key reason why many people seem to care about RSs is money. Another reason why data scientists specifically should care about RSs is that it is a true data science problem. More specifically, Movie recommendation is the most widely used application coupled with online multimedia platforms which aim to help customers to access preferred movies intelligently from a huge movie library. Hence we aim to build a movie recommendation system that implements various recommendation approaches based on clustering and filtering techniques.

II. PROBLEM STATEMENT

This project is based on a recommender system that recommends movies to users based on various aspects. This system will provide more precise results as compared to the existing systems. The existing system works on individual users' rating. This may be sometimes less productive for the users who have different taste from the recommendations shown by the system. This system calculates the similarities between different users and then recommends the movie to

 ISSN: 2321-8169
305 – 310

them as per the ratings given by those different users of similar tastes. This will provide a precise recommendation to the user.

III. LITERATURE REVIEW

There are two important approaches in the field of RSs.One is Personalized and another one is Non-Personalized. Personalized RSs are used by E-commerce website which suggests products to their customers mainly based on their past purchase history. On the other hand,Non-personalized RSs suggests products to their customers based on the reviews of the products given by other customers who may have bought that product.

Recommender systems can also be functionally classified based on what type of algorithm is being used.

A. Content based filtering:

This filtering technique is based on description of the item and a profile of the user's preference. These algorithms recommend items that are similar to those that a user liked in the past. A user profile is built to indicate the type of item this user likes.

B. Collaborative filtering(CF):

This is sub divided as item based CF and User based CF.

In user based CF,the system finds users whose past behavior pattern is similar to that of the current user and use their ratings on other products to predict what the current user may like. Thismethod suffers from scalability problems as the user base grows.

In Item based CF,the system uses similarities between the rating patterns of items. If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items. This method is similar to content-based filtering but here item similarity is deduced from user preference patterns rather than extracted from item data.

C. Hybrid recommender system based filtering:

This is a combination of Content based and Collaborative filtering. These systems are used to eliminate some of the drawbacks of recommender systems such as cold start and sparsity problem.

IV. PROJECT DESCRIPTION

A. Overview of the project:

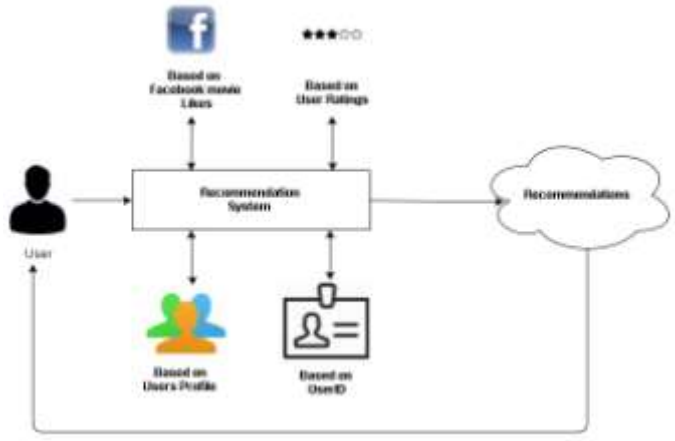

Figure 1: Overview of the project

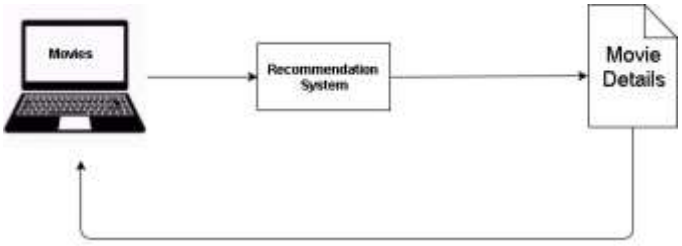

Figure 2: Project Flow

We propose a RS for movies which provides a variety of options using which the user can get recommendations. The diagrams above give an abstract overall view of the system. As shown in figure 1, the recommendation engine corresponding to the option chosen by the user runs the particular algorithm and generates recommendations for the user. As shown in figure 2, two more modules are developed and integrated into the recommendation system. One module displays the popular movies to the user and the other module provides details of the movies that reside on the user's machine.

B. Dataset:

We performed an experiment on a subset of movie rating data collected from the MovieLens web-based recommender. The dataset contained 100,000 ratings from 943 users and 1,682 movies, with each user rating at least 20 items. We divide the dataset into a training set and a test data set. The detailed description of the files used in the dataset are available at https://grouplens.org/datasets/movielens/100k/.

C. Survey statistics:

A survey was conducted before developing some modules of this project. The Google Form consisted of

ISSN: 2321-8169
305 – 310

questions to get to know the preferences, likes, and choices of people in general. The form was filled by around 150 people belonging to various age group, gender, and occupation. The graphs given below are the output of the preferences submitted by the users.

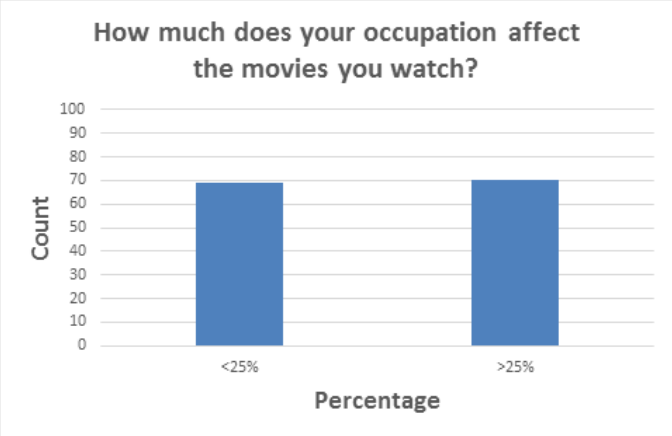

Figure 3: Survey Statistics-1

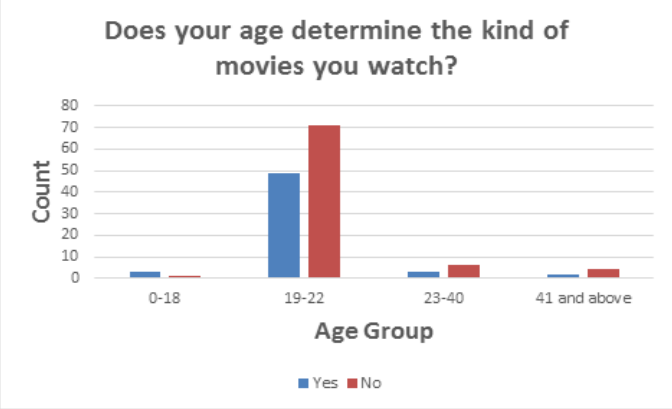

Figure 4: Survey Statistics-2

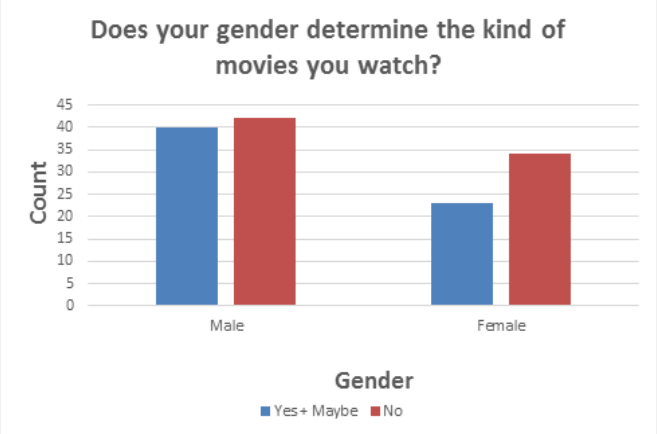

Figure 5: Survey Statistics-3

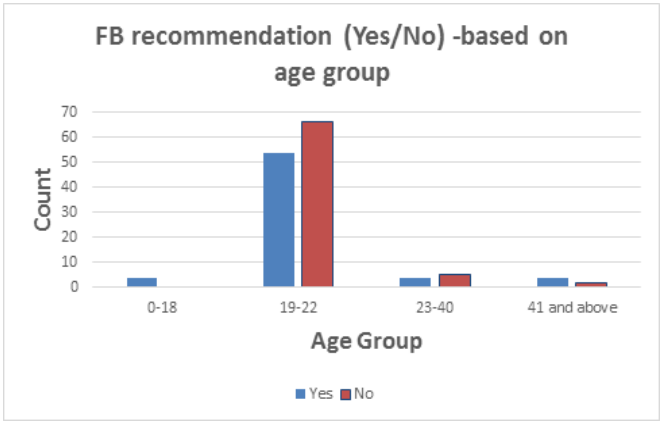

Figure 6: Survey Statistics-4

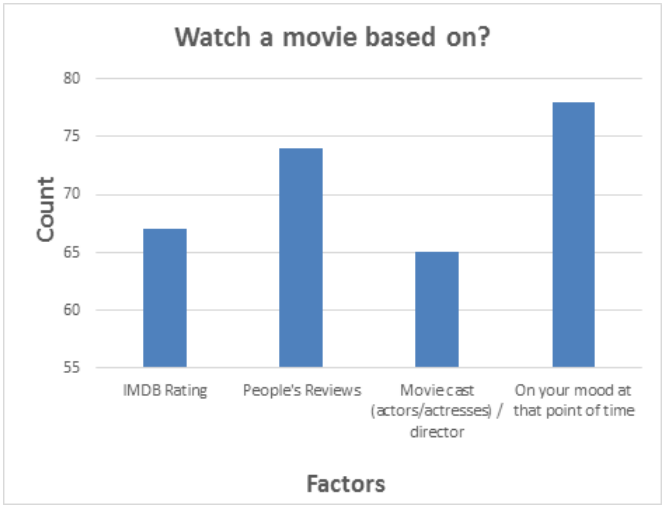

Figure 7: Survey Statistics-5

D. Module description:

1) A Popular Movies and Genre based on dataset: GroupLens Research has collected and made available rating data sets from the MovieLens website (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. As described earlier Dataset consists of Movies belonging to various Genres and have been rated by many users. So, based on that we perform analysis on the dataset to find:

a) Most Popular Movie: A movie with maximum value of sum of its ratings is considered as the most popular movie.

b) Most Watched Movie: A movie which has been rated maximum number of times is considered to be the most watched movie. Because a rating assigned to a movie indicates the user has watched that movie.

c) Most Watched Genre: We sort movies according to their genres and then find the movies with a maximum number of ratings and genre belonging to these movies is considered as the most watched genre.

d) Most Popular Movie for each Genre: We sort by genre and then the movie with maximum value for the sum of ratings is considered as the most popular movie for that particular genre.

2) Facebook based Recommendation Module: Whenever a user uses traditional recommendation service for the first time, the system has no information about the movies the user has watched and liked/disliked. Despite all this, the system still has to provide good recommendations in order to retain the interest of the user. The simplest or common solution is to recommend popular movies. But this solution is not at all personalized and users might lose interest in using such systems.

The proposed system will help to solve this problem and also provide personalized recommendations to the user. The recommendation engine for this module fetches the movies liked by the user. These movies are forwarded toThe Movie Database (TMDb) API and similar movies are recommended as the final output. The module is implemented in Python and makes use of Facebook Software Development Kit (SDK) for PHP which helps to easily integrate Facebook login and make requests to the Graph API.

3) Recommendations based on basic User details (User-based): Some users are apprehensive about sharing their Facebook details. Such users can provide their basic details such as age, gender, and occupation to the recommendation engine of this module. In the pre-processing part, the users of the dataset are assigned to different clusters based on the similarities between them. The neighborhood of the new user (i.e. similar users) is found. The movies watched by such users of the neighborhood are then recommended to this new user.

4) Recommendations based on User's Ratings (Item-based): The recommendation engine for this module takes into consideration the user's past ratings (for existing user) and also two new movie ratings. The two new movie ratings are used to solve the cold start problem for a new user. The similarity between the movies rated by the user and the movies from the dataset is calculated using similarity measure. Here the similarities between the items (i.e. movies) are considered as compared to similarities between the users mentioned in the above module.

5) Recommendations for an existing User based on User-ID: This module is especially for an existing user who has already rated some movies before. The user just needs to enter his User-ID as input. The recommendation engine follows the hybrid approach. In this approach, it makes use of a combination of User-based and Item-based approaches. This hybrid approach overcomes the shortcomings of the individual approaches. A mixed type of hybrid recommendations engine is implemented in this module. The recommendations from both the approach are combined to give the final recommendation to the user.

6) Details of movies available on the host machine: Users usually tend to have a lot of movies on their system which they get from their friends or download from various sources. Every time they wish to see a movie from that list they have to Google about that movie for details like movie's The Internet Movie Database (IMDb) rating, genre, actors, plot etc. The above task is doable if the user has only a few movies on his system, however, it becomes tedious to do the same for a large set of movies. Our module handles this problem very easily providing the user with the relevant details of the movies in the directory given as input by the user.

V. METHODOLOGY AND IMPLEMENTATION

A. Displaying popular movies and genre:

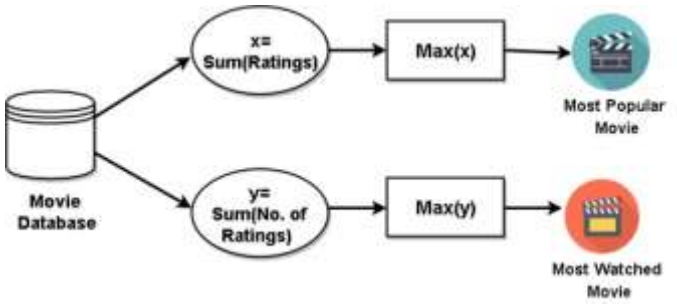

Figure 8: Displaying Popular Movies and Genre

In this, no input is taken from the user. The recommender engine for this module processes and analyses the ml-100k data set, movie details dataset to find the following 4 things: Most Watched Movie, Most Popular Genre and Movie and also Most Popular Movie for a Genre.

B. Displaying popular movies and genre:

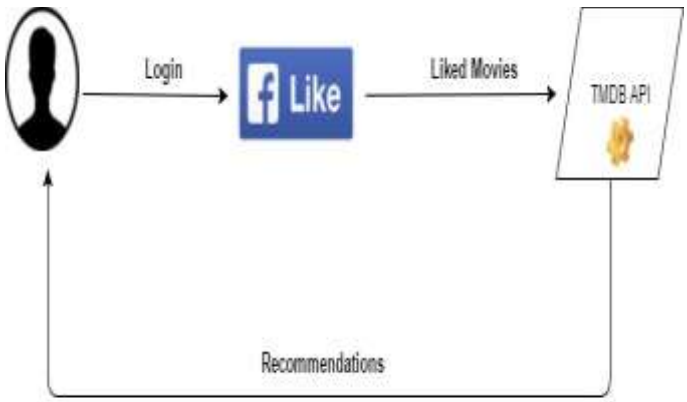

Figure 9: Recommendation using Facebook data

 ISSN: 2321-8169
305 – 310

In these section, User's Facebook liked movies are used to recommend similar movies. The recommendation engine for this module uses Facebook SDK for PHP. The Facebook SDK for PHP enables us to easily integrate Facebook login and make requests to the Graph API.

C. Recommendation using Clustering method:

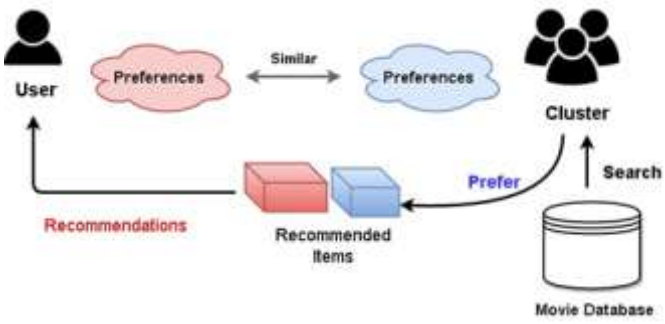

Figure 10:Recommendation using Clustering Method

Proposed system gives a choice to the user wherein the user can get recommendations by just providing the system with their basic details such as age, gender, and occupation. The concerned recommendation engine already runs the clustering algorithm on the available user dataset. The clustering algorithm clusters similar users into one group. The basis for clustering the users is the movies they rated and how well they rated it.

After the user inputs his/her details, the users with the same basic details are found from among the dataset. After finding such users, the cluster centroid (which is basically the mean ratings of the users belonging to the particular cluster) for each of these users is found. The average of all these ratings is then calculated and the movies are arranged from higher to lower order of preferences. The top n preferences are then suggested to the user where n is the number of recommendations that the user requests.

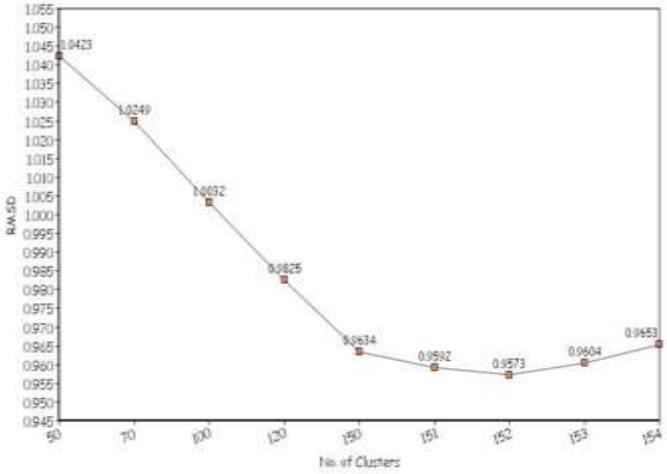

Figure 11:Graph showing RMSD Vs Number of Clusters

The clustering algorithm used here is k-means which partition the data set into k clusters. We divided our dataset into 80% as training data and 20% test data. We calculated the RMSD (Root Mean Square Deviation) for various values of k. We got the lowest RMSD for k=152.

D. Recommendation using Similarity Measure:

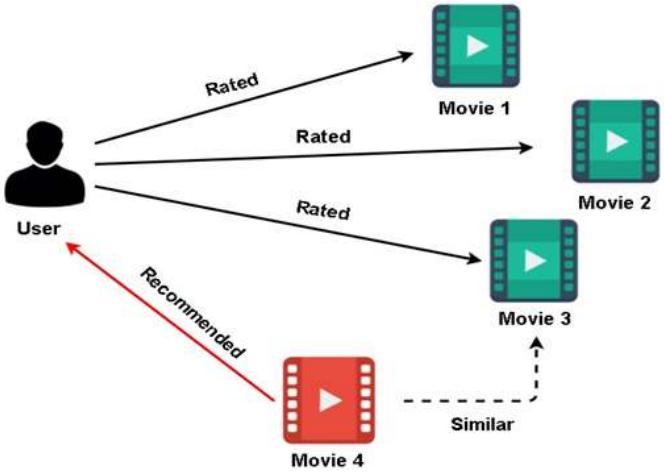

Figure 12:Recommendation using Similarity Measure

We have data of users and their ratings to various movies. Based on these ratings, a rating vector for each movie is formed. Based on the movies that a particular user has liked, similar movies are found using similarity measure. Various similarity measures were considered for finding the similarity between different movies. Jaccard Similarity doesn't take ratings into considerations. So, it was discarded. Cosine similarity was tried next but it also faced the issue of ratings not being normalized. To overcome issues in the both abovementioned methods, we used Pearson Similarity. It is also called adjusted cosine similarity.

E. Recommendation using Hybrid Approach:

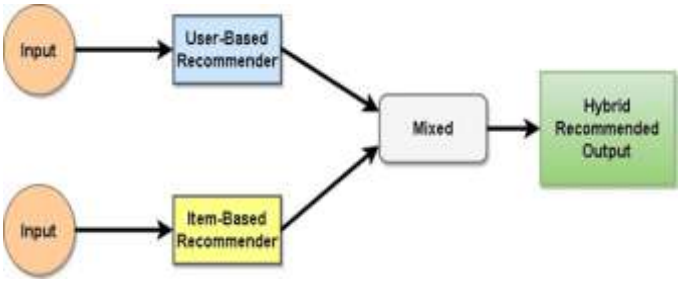

Figure 13:Hybrid Recommendation Approach

Figure 13 shows the general flow for the recommender engine. For this type of recommendation, we have considered only the user-id. This module makes use of both user-based and item-based recommendation engine. After the user inputs

 ISSN: 2321-8169
305 – 310

the id, all the information pertaining to him like his basic details (age, gender, occupation) can be retrieved from the dataset. The user rated movies can also be obtained in the same way. Clustering is performed on the complete dataset and the user-based recommendations are generated using this way. Using the movies that the user has rated, movie vector is formed and similar movies are found using adjusted-cosine similarity computation method. The top recommendations obtained from both the engines are then mixed together to give final recommendations to the user. Making use of both the systems overcomes the shortcomings of the other and help us solve the cold start problem.

F. Information about movies on local machine:

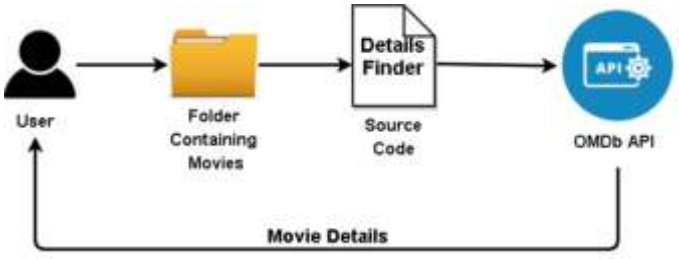

Figure 14:Local Movie Information from OMDb

This module is aimed at saving time and energy of the user. The user just needs to input the directory name containing the movies whose details the user wants to see. Once the directory is entered the code processes the folder/file names in the directory to extract the movie names from it. These movie names are then used to find the movie details from The Open Movie Database(OMDb) API.

The OMDb is a database of movie details. It takes the movie name as input and provides the movie details as output. The user can go through these details and then decide on which movie he would like to watch from the long list of movies available in his movies directory.

VI. CONCLUSION AND FUTURE SCOPE

A recommendation system has been implemented based on hybrid Approach as well as through Social media platform. We have tried to combine the existing algorithms for recommendation to come up with a hybrid one. It improves the performance by overcoming the drawbacks of traditional recommendation systems. The hybrid recommendation engine is a competent system to recommend Movies for users, whereas the other recommender algorithms are quite slow with inaccuracies. Our work indicates that the correct application of the item information can improve the recommendation performance.

Our approach can be extended to various domains to recommend books, music, etc. There are some cases in which collaborative and content-based methods are not useful in obtaining meaningful recommendations because of the high degree of complexity and constraints in the item space. In such cases, knowledge-based recommender systems are particularly useful. Multi-criteria recommendation approaches are undergoing significant developments. The exploitation of multi-criteria scores, which contain contextual information, would be useful in improving recommendation quality in the future. As for future work, our approach can be improved to deal with higher dimensionality and sparsity issues in practical environment and More effective data reduction algorithms can be coupled with clustering-based CF. Furthermore, how the variation in the number of clusters will influence the movie recommendation scalability and reliability can be studied. To generate high personalized movie recommendations, other features of users, such as tags, context, and the web of trust should be considered in future studies.

REFERENCES

[1] Python Contributors, Available at https://docs.python.org/3/

[2] The OMDb (The Open Movie Database) API Contributors, Available at http://www.omdbAPI.com/

[3] Stackoverflow Contributors, Available at https://stackoverflow.com/

[4] The TMDb (The Movie Database) API Contibutors, Available at https://www.themoviedb.org/?language=en

[5] Facebook for Developers Contributors, Available at https://developers.facebook.com/docs/graph-api

[6] Tutorial Point Contributors, Available at https://www.tutorialspoint.com/

[7] Quora Contributors, Available at https://www.quora.com/

[8] MovieLens Contributors, Available at https://movielens.org/home

[9] Wikipedia Contributors, Available at https://en.wikipedia.org/