Popularity Based Recommendation System

Keshetti Sreekala

Abstract: ***A Recommendation engine recommends the most relevant items to the user by using different algorithms to filter the data. A Recommendation system is more useful in the context of data extraction relating to applications of big data and machine learning. As the name indicates Popularity based recommendation system works with the current vogue. It basically uses the items which are in swing at present. This is the most basic recommendation system which provides generalized recommendation to every user depending on the popularity. Whatever is more popular among the general public that is more likely to be recommended to new customers. The generalized recommendation not personalized is based on the count. In this paper I am going to use a class that we created which includes the methods to create recommendations and to recommend the item to the user. Next I will load the data of Comma Separated Value (CSV). After that sort the sound name based on the how many users have listened to the sound name. After the collection of data code splits the dataset into training and the test dataset using 80–20 ratio. This creates an instance of popularity based recommenders class. At last I will use the popularity model to make the predictions.***

Keywords: popularity, recommendation, sound name, prediction, listen, dataset, filter, trend.

I. INTRODUCTION

Recommender engines (REs) also known as recommender systems are becoming very popular software tools as they are able to provide good recommendation about a product to a user. The suggestions offered are focused at helping their users in various decision making processes such as what products to buy, which music to listen and whose profiles to browse, or what news to read. A recommendation engine recommends the most relevant data to the users by filtering the data using different algorithms. Recommended systems are useful in so many areas like news, movies, research articles, music, books, search queries, social tags, and products in general. As the name implies Popularity based recommendation system works with the ongoing trend. It basically uses the items which are more popular at present. The item that is more popular among the general public, is more likely to be recommended to new customers. But this is more generalized recommendation rather than personalized. Recommendation system is more useful application plug-in that is being used by most of the online service providers to understand and serve the needs of online users. Recommender System tries to predict the interest of users in terms of products. Whenever we buy a new product it is usual tendency to take feedback of our friends or relatives who are already using that product. But now with the increased customers of online shopping as well as busy life schedules people are more interested in buying the products online. This is where a recommendation system is more useful as it suggests which product is better to choose. Recommender System identifies products that best fits user choices. These techniques are beneficial to customers as well as vendors as they enable the customers to quickly find what they need and the vendors to promote their products and sales. Because of the popularity of recommendation system and the benefits that could be accomplished by using these algorithms, it has become a very interesting problem for researchers also. Hence this kind of systems became very active since the last 30 years. At present it has become a great challenge to provide recommendations to people from the large data available on the internet. E-commerce giants like Amazon, EBay provide personalized recommendations to users based on their taste and history but Machine Learning (ML) and deep learning techniques can provide more appropriate recommendations.

II. RELATED WORK

With the increased use of popularity based recommendation system researchers are more interested in developing recommender systems which are very useful while we are trying to buy any product through online. In the past different factors have been identified and explained to increase the reliability of recommender system. Recommender systems are tools used for filtering and sorting items and information. They are efficient tools that overcome the information overload, by providing users with the most relevant information by their interest. These systems are usually based on the user preference and rating. The ratings can either be acquired explicitly by filling up form, providing ratings or implicitly. Since the goal of a recommender system is to give or suggest a best recommendation to the user. This suggestion can be made by identifying the category of the product and based on some of the characteristics of the product's data available. Apart from this the recommender system may also has access to some specific attributes that are related to user or product.

Following are the few techniques investigated by me and motivated me to develop popularity based recommendation system. John O'Donovan et. al.[1], have taken trust as the percentage of correct predictions that a profile has made in general (profile-level trust) or with respect to a particular item (item-level trust). They have described a number of ways in which these different types of trust values might be incorporated into a standard popularity model algorithm and evaluated each against a tried-and-test benchmark approach and on a standard data-set.

Revised Manuscript Received on February 15, 2020.
***** Correspondence Author
Keshetti Sreekala*, Department of CSE, MGIT, Hyderabad, India. Email: ksrikala_cse@mgit.ac.in

This decreases the prediction error only by 22%. Mohsen Jamali et. al. [2] proposed Trust Walker; it is a random walk model for combining trust-based and popularity model recommendation. Collaborative recommender system cannot make recommendations for so-called cold start users that have rated only a very small number of items. The random walk model allows us to measure and define the confidence of a recommendation. Trust Walker combines the trust-based and the popularity model filtering approach to recommendation. Trust Walker allows us to define and to measure the confidence of a recommendation. This model is suffering from cold start issue and identification of domain. Paul Resnick et. al. [3] proposed an idea of "influence limiter algorithm" for recommender system. This algorithm prevents attack that has irrelevant result for my search. This algorithm limits the number of content that an attacker can modify. The paper presented by Portugal et. al.[4] presents a systematic review of the use of Machine Learning in recommender systems. They analyzed primary studies and classified recommender systems in different categories: popularity model, content-based and neighbor-based of content-based filtering, neighborhood-based and model-based of collaborative filtering, and hybrid filtering. This work helps developers to recognize the algorithms, their types, and trends in the use of specific algorithms. It also offers latest evaluation metrics and categorizes the algorithms based on these metrics. But this algorithm works only with sparse data. Mustanser Ali et. al. [5] proposed a new recommender system that is based on deep learning to overcome some of the existing limitations. This is Popularity based recommendation system that recommends items by taking into the listen count of the users into account. But sometimes assumptions may not yield the accurate results. These are certain challenges that motivated me to develop a recommendation system that depends on external community data and generate recommendations.

III. DESIGN AND WORKING OF POPULARITY BASED RECOMMENDATION SYSTEM

Architecture of Popularity Based Recommendation System

Steps involved in building a popularity based recommendation system are as follows:
1) Downloading the required data set. 2) Build a popularity based model and retrieve the scores. 3) Find the scores obtained. 4) Based on the computed scores, make the appropriate recommendations according to the input. First I will import the numpy and the pandas which I am going to use a lot and the class that I have created which includes the methods like create that basically creates the recommendations and the recommend method that recommends the items to the user. After that I will load the data of comma separated value (CSV).Later I will sort the soundname based on the how many users have listened to the soundname. In next step code splits the dataset into training dataset and test dataset using 80–20 ratio. This facilitates to create an instance of popularity based recommenders class. Next I will use the popularity model to make the predictions. The overall system design has the following phases:
A) Data set selection B) Training the Popularity Based Recommendation system C) Recommendations

A. Data set selection:

There are many dataset available on internet for building recommendation such as help reviews dataset, movielens dataset .This paper mainly deals with soundname recommender system .The dataset consists of one Lakh popular music tracks. Dataset consisting of user id, song id, listen count, created by, and artist name.

B. Training the Popularity Based Recommendation system

Code splits the dataset into training and the test dataset using 80–20 ratio. In memory-based methods the system predicts based on the similarities of the matrix which is pre-computed. I will collect the tags that are assigned to each song which are given by different users, add the name of the song and each song also contains a metadata column..

C. Recommendations

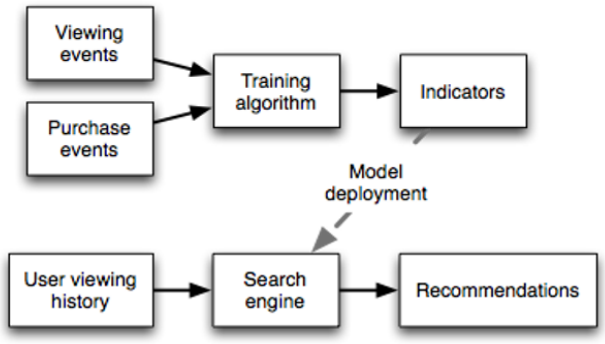

Figure3.1: Architecture of Popularity Based recommendation system

As shown in the above Figure 3.1, describes that Recommendation is created by splitting the dataset into training as well as testing dataset. Recommendation is done by using a python program that finds the most popular songs in dataset and selects the song for which users have reacted positively. Same songs are recommended to all the users.

Recommendation Modules

The recommendation module consists three sections:
• Mean algorithm
• Recommendations based on the Listencount of the user.
• Calculating the score and listing the Recommendations.

Mean Algorithm:

This algorithm is the usual non-personalized recommendation algorithm that considers user interest that is present in the dataset. The Mean algorithm also referred as the item average algorithm or the POP algorithm. This algorithm is used in the popularity prediction techniques (Breese et. al. [6]). Whenever the prediction is needed for an item/product it is calculated based on the average of polls of all users of that item/product. The mean value of an item i is defined as:

$$\overline{p_i} = \frac{1}{|P_i|} \sum_{i \in P_i} v_{ij}$$

Where p_i is the set of users who have rated/Listened item i,

and v_{ij} is the user vote for item i.

Recommendations based on the Listencount of the user:

The aim of the recommender system [7] is to give a better and genuine service by creating an esteemed bond between the user and the site. Because of the establishment of the bond, the user will be returning back to the interested sites and repays. (Konstan et al.,[8][9]). This is the reason for finding three different kinds of non-personalized recommendations over the information page for all the songs and this is based on an average of other user's sentiments that are already collected about the song. The users can look at the book details page to know the about the statistical data. This page is constructed based on the listencount and rankings given to the songs by the viewers. The user will also be able to look at the ranking/counting of any song as well as the number of users rated/ranked that particular song.

Calculating the score and listing the Recommendations:

This constitutes the stage where another model further ranks and scores the candidates usually on a scale of 10. For instance, in the case of YouTube, it uses a plenty of lineaments that describe the video and user who assigns a score to the video/Song. This is how the recommendation system of YouTube works. The songs that are having high score are displayed to the user. Based on the rank given by the user and on the sound name the recommendations will be provided to the user.

Figure 3.2 shows the simplest representation of a user's interaction with the system. It shows the relationship between the user and the different use cases in which the user is involved. This use case diagram helps in identifying the different types of users of a system and the different use cases. The circles or ellipses are used to represent the use cases.

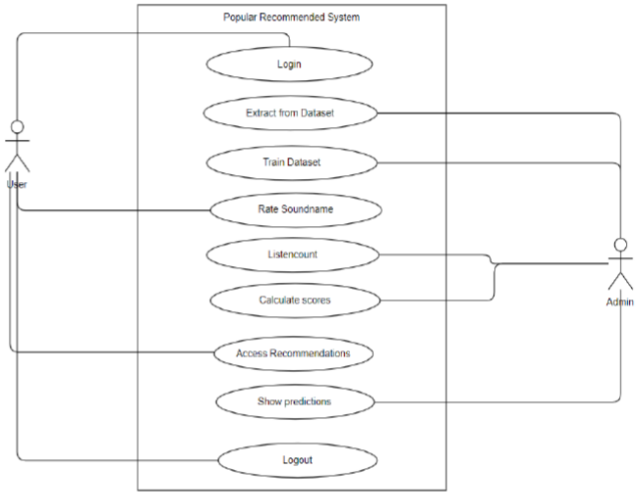

Figure3.2: Use case diagram of Popularity Based Recommendation System

The user has the option to rate/listen the soundname and the admin has to count the soundname and calculate the scores. The resultant top N recommendations are given out as a result and now user can access the recommendations [10].

Below Figure 3.3 shows the class diagram of the popularity based recommendation system. There are four main classes used in the system.

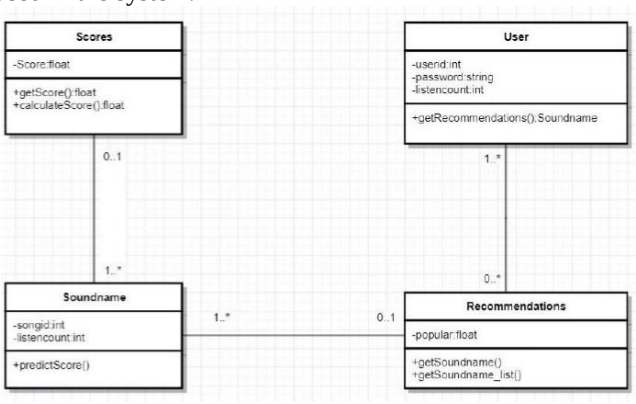

Figure3.3: Class diagram of Popularity Based Recommendation System

The user class contains a list of userid, password and listencount of each user and based on that listencount he/she will get the recommendations. The Soundname class contains the songid and also the listencount of user and based on that score will be predicted. The score class contains the calculated scores. The Recommendation class contains the popular recommendations or top N popular Recommendations to the user.

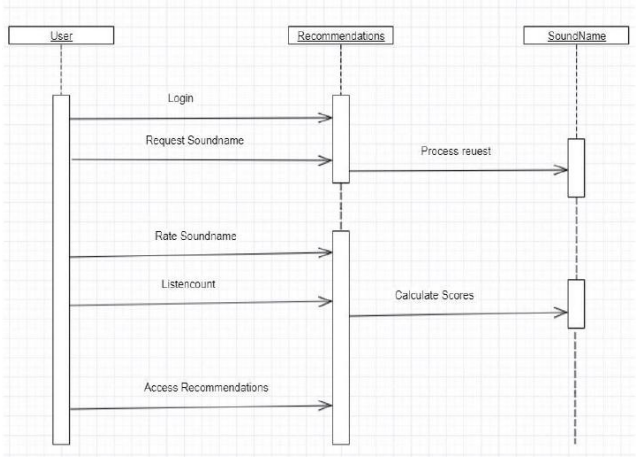

Figure3.4: Sequence diagram of Popularity Based Recommendation System

The above figure 3.4 shows that, the user tries to log in to the application. Based on the listencount of the user, the scores will be calculated and after calculating the scores will be the popular top N recommendations that the user can access.

Figure 3.5 shows the activity involved in popularity based recommendation system. It makes us to understand the flow of the process involved in the system.

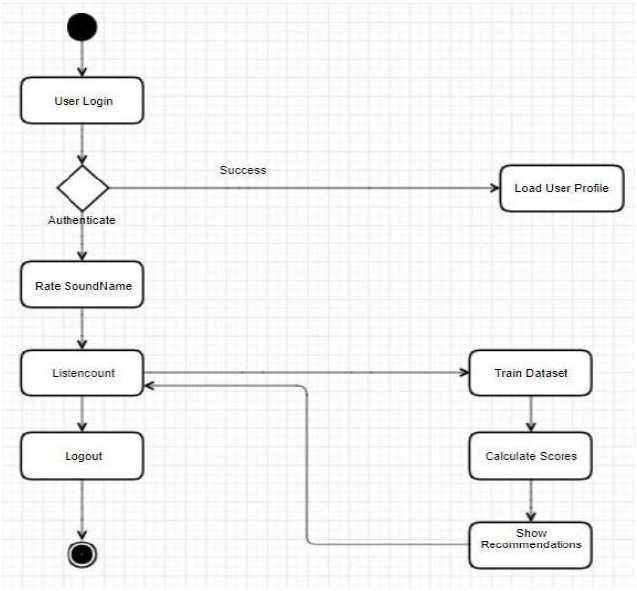

Figure 3.5: Activity diagram of Popularity Based Recommendation System

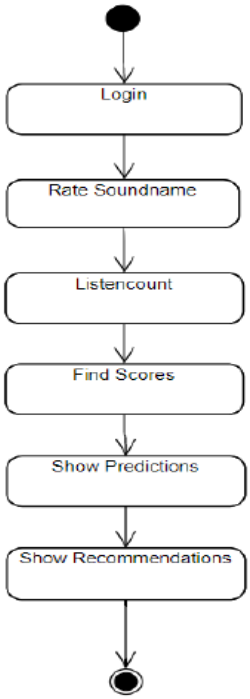

Figure 3.6: State diagram of Popularity Based Recommendation System

Figure 3.6 shows the state diagram of Popularity Based Recommendation System that describes the different states of the system. The user can rate/listen the soundname. The scores are calculated and then recommendations are displayed.

IV. EXPERIMENTAL SETUP AND RESULTS

In order to carry out the desired functionality I have used a system 4GB RAM, 20GB hard disk and a processor of speed 2GHZ with windows 10 operating system. The softwares used are anaconda package manager, jupter notebook IDE, Numpy, pandas, Scikit Learn Library packages and python programming language.

OVERVIEW: In this section, I will provide an outline of trial runs of the system and also discuss about the list of features that are used to carry out these trial runs. As the creation of data set is time consuming and there are many datasets available on the internet I have downloaded the dataset Free Music Archive [11] and used it for performing trial runs. The dataset consists of one Hundred Thousand popular music tracks. Few of the fields in the dataset are user id, song id, listen count, created by, and artist name. As discussed in section 3 once the user selects the song of his/her choice from the dataset this data and previous history are fed as input to the recommendations system. Then mean algorithm is applied on this data. For any product or item, the average rank of an item i is calculated by using the following formula:

$$\overline{p_i} = \frac{1}{|P_i|} \sum_{i \in P_i} v_{ij}$$

Where p_i is the set of users who have rated/Listened item i, and $_{vi\,j}$ is the user vote for item i.
The following data in Figure 4.1 is the given input to the system. It shows that from the given data set I have considered the first 15,000 rows.

```
song_file='userdata.csv'
song_df=pandas.read_csv(song_file)
song_df = song_df.head(15000)
song_df.head()
```

Figure 4.1 input for the Popularity Based Recommendation System

In the below figure 4.2 it generates a key error because I have chosen user id 50567. This is beyond the row number I have considered. Hence it generates an error.

```
pr.recommend(users[50567])
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-14-83c860bd7139> in <module>
----> 1 pr.recommend(users[50567])

~\Anaconda3\lib\site-packages\pandas\core\series.py in __getitem__(self, key)
    866         key = com.apply_if_callable(key, self)
    867         try:
--> 868             result = self.index.get_value(self, key)
    869
    870             if not is_scalar(result):

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_value(self, series, key)
   4373         try:
   4374             return self._engine.get_value(s, k,
-> 4375                                           tz=getattr(series.dtype, 'tz', None))
   4376         except KeyError as e1:
   4377             if len(self) > 0 and (self.holds_integer() or self.is_boolean()):

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 50567
```

Figure 4.2 output generated for the user id that is beyond the row number.

When the recommend class is used as a black box to train the system first we need to create an instance of a recommender class and feed it with the training data. Based on the popularity of each song, we need to create a recommender that accepts user_id as the input which then outputs a list that contains the recommended soundname of that user. This whole activity is illustrated in the below figure 4.3

```
pr = Recommend.Popularity_Recommender()
pr.create(train_data, 'user_id', 'SoundName')
```

```
pr.recommend(users[5])
```

	user_id	SoundName	score	Rank
80	485d9e2e6a01fe63	ssss	253	1.0
46	485d9e2e6a01fe63	love	186	2.0
19	485d9e2e6a01fe63	cnk	160	3.0
14	485d9e2e6a01fe63	bala bala	158	4.0
38	485d9e2e6a01fe63	jani	157	5.0
92	485d9e2e6a01fe63	vadivel comedy	155	6.0
58	485d9e2e6a01fe63	nasir	153	7.0
61	485d9e2e6a01fe63	ninna sneha	152	8.0
23	485d9e2e6a01fe63	dil	151	9.0
44	485d9e2e6a01fe63	kottoddu babai	151	10.0
55	485d9e2e6a01fe63	my fav by sk	151	11.0
94	485d9e2e6a01fe63	xxxtentacion	151	12.0
47	485d9e2e6a01fe63	maacho	150	13.0
54	485d9e2e6a01fe63	moi	149	14.0
51	485d9e2e6a01fe63	manass	148	15.0

Figure 4.3: Recommendation list generated by popularity based recommendation system

As shown in figure 4.3 the method recommend gives the recommendation to that user which is passed as an argument to the method. Thus it returns the list of popular songs for that user but since it is based on popularity the recommendation for the specified user is generated based on popularity. Thus the required functionality of the popularity based recommendation system is achieved.
Following table 4.1 shows the description of explicit feedbacks.

Table 4.1: Description of explicit feedbacks

feedback	Description
Userlike	An integer between 1 to 5 depending on how much the user liked the song
Relevant	An integer between 1 to 5 depending on the raga style
Novelty	An integer between 1 to 5 representing how much the song is related to western music.
Listenornot	A Boolean value whether to listen (1) the song or not (0).
classes	A string representing the classes for which the song (for example: ,album, movie) as the song may belong to multiple ',' serves as delimiter.

Below is the table 4.2 which shows about the description of implicit feedbacks.

Table 4.2: Description of implicit feedbacks

Feedback	*Description*
TimeOnMouse	Number of milliseconds the user spent on moving the mouse.
TimeOnPage	Number of milliseconds the user spent on a page or news article.
EventOnScroll	Number of clicks on the scroll bars.
ClickOnWindow	Number of clicks inside browser window but not on scroll bars.
TimeOnHScroll	Number of milliseconds the user spent on using horizontal scroll.
TimeOnVScroll	Number of milliseconds the user spent on using vertical scroll.
NumOfPageUp	Number of pages the user scrolled up.
NumOfPageDown	Number of pages the user scrolled down.
MSecForPageUp	Number of milliseconds the user spent on scrolling a page up.
MSecForPageDown	Number of milliseconds the user spent scrolling a page down.
NumOfUpArrow	Number of clicks on up arrow key.
NumOfDownArrow	Number of clicks on down arrow key.
MSecForUpArrow	Number of milliseconds the user spent on up arrow key.
MSecForDownArrow	Number of milliseconds the user spent on down arrow key.

The variables given under the column feedback of table 4.1 served as the set of features for this model. I have used all the features except the userlike as that is to be calculated based on the remaining features. These features help in predicting user ratings. Time on mouse and Time on page features of implicit feedback are considered in calculating the userlike attribute.
Below is the table4.3 which shows explicit feedback statistics.

Table 4.3: explicit feedback statistics.

feedback	min	max	mean
userlike	1	5	3.76
Relevant	1	5	4.20
Novelty	1	5	3.87
Listenornot	1	0	0.78

Statistics on implicit feedback are show in below table 4.4.

Table 4.4: Statistics on implicit feedback.

Feedback	*min*	*max*	*Mean*	*Variance*
TimeOnMouse	0	320611	1976.4	29049887.3
TimeOnPage	0	5138378	71430.3	18683642234
EventOnScroll	0	160	0.92	13.7
ClickOnWindow	0	81	0.87	5.5
TimeOnHScroll	0	0	0	0
TimeOnVScroll	0	0	0	0
NumOfPageUp	0	19	0.1	0.67
NumOfPageDown	0	19	0.12	0.86
MSecForPageUp	0	5978	17.3	32784.86
MSecForPageDown	0	7391	22.46	50489.13
NumOfUpArrow	0	23	0.08	0.52
NumOfDownArrow	0	91	0.96	18.24
MSecForUpArrow	0	7378	23.26	46440.9
MSecForDownArrow	0	17743	175.1	641665.3

As discussed in above table4.1, table4.2, table 4.3 and table 4.4 the implicit and explicit feedback features are calculated, based on these calculations the method recommend gives the recommendation As so many features are taken into account while calculating the mean the popularity based recommendation system returns the list of popular songs for that user based on popularity. Thus the required functionality of the popularity based recommendation system is achieved.

V. DISCUSSION

In the existing system, when we open the app it directly shows the products based on the popularity but it cannot show all the products. Where as in the system we have developed it directly shows all the products available at first and then when we select the product it shows the popularity of the product. So in the existing system user may not be able to find the product of his choice but in the system we proposed it is possible. Also existing system takes more time as it has to search and place each and every product ‘s popularity but in our system search time is minimized as it shows the popularity of only the selected product. Thus the popularity based recommendation system helps in predicting the number of users used the selected product. The system also generates a rating based on the average rating of the users of that product. The computation involved is very less Hence the proposed popularity based recommendation system works fast and uses less amount of memory.

VI. CONCLUSIONS

Recommendation structures are proving to be a useful device for addressing a part of the records overload phenomenon from the internet. The main objective is that it is the easiest way to build a recommendation system that is popularity based, simply over all the products that are popular, So how to identify popular products, which could be identified by which are all the products that are bought most. Based on the customer interest and need if we can recommend suitable product for them we can attract the customer by creating a positive opinion so that the user may also visit to the site number of times. This is one of the key aspect of making any businesses incremental. This leads to a smart and intelligent business which we can achieve with the help of recommendation engines, and by studying the past behaviour of their users. The proposed approach is the best solution as it takes the customer interest into account and then it generates the popular products.

REFERENCES

1. John O’ Donovan, Barry smyth, Trust in recommender systems, Proceedings of 10th international conference on intelligent user interfaces, January 10-13, 2005, San Diego, Califoriia, USA.
2. Mohsen Jamali, Martin Ester, Trust Walker: a random walk model for combining trust based and popularity model recommendation, Proceedings of the 15th ACM SIGKDD 273 Parul, Kavita Khanna international conference on knowledge discovery and data mining, June 28- July 2.
3. Paul Resnick, the influence limiter: provably manipulation resistant recommender systems, Recsys 07, October 19-20, 2007, ACM, Minneapolis, Minnesota, USA
4. Portugal, Implementation of item and content based collaborative filtering, popularity model techniques based on ratings average for recommender systems, Universoity of Mumbai 2013.
5. Francesco Ricci, Lior Rokach and Bracha Shapira, Introduction to recommender systems hand book, 2011.
6. Breese, J., Heckerman, D., and Kadie, C. (May, 1998). An experimental comparison of collaborative filtering methods. Technical Report MSR-TR-98-12, Microsoft Research, Redmond, WA.
7. Resnick, P. and Varian, H. (1997). Recommender systems. Communications of the ACM, 40(3):56-58.
8. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., and Riedl, J. 1997Applying Collaborative Filtering to Usenet News. Communications of the ACM 40(3):77-87
9. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon,L.R., and Riedl, J. 1997Applying Collaborative Filtering to Usenet News. Communications of the ACM 40(3):77-87
10. L.Terveen, Hill, W., Amenta, B., McDconald, D., and Creter, J. (1997). PHOAKS: A system for sharing recommendations. Communications of the ACM, 40(3):59-62.
11. Defferrard, Michaël; Benzi, Kirell; Vandergheynst, Pierre; Bresson, Xavier (6 December 2016). MA: A Dataset For Music Analysis arXiv:1612.01840 [cs.SD].

AUTHORS PROFILE

Dr Keshetti Sreekala is working as an assistant professor in the Department of computer science and engineering at Mahatma Gandhi Institute of Technology, Gandipet, Hyderabad, Telangana State, India. She received her Ph.D. in Computer Science and M.Tech degrees from Jawaharlal Nehru Technological University, Hyderabad. Her research interest spans wide coverage and includes Machine Learning, Internet of Things and Grid computing. Much of her work has been on Grid Computing. She has published her works in IEEE conferences and Scopus indexed journals. She also guides students who are working on Machine Learning and Internet of Things based problems