

**USER-ITEM BASED HYBRID RECOMMENDATION
SYSTEM BY EMPLOYING MAHOUT FRAMEWORK**

Project submitted to
**FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY**

In partial fulfilment of the requirements for the degree of
MASTER OF COMPUTER APPLICATIONS, 2018

BY

Sutanu Paul

Examination Roll: MCA186017

Registration No: 133679 of 2015-2016

Under the guidance of

Dr. Dipankar das

Assistant Professor, Department of Computer Science &
Engineering

Jadavpur University

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

TO WHOM IT MAY CONCERN

I hereby recommend that the project entitled “USER-ITEM BASED HYBRID RECOMMENDATION SYSTEM BY EMPLOYING MAHOUT FRAMEWORK” prepared under my supervision and guidance at Jadavpur University, Kolkata by SUTANU PAUL Reg. No. 133679 of 2015 – 16, Class Roll No. 001510503017 of 2015-16), may be accepted in partial fulfillment for the degree of Master of Computer Applications in the Faculty of Engineering and Technology, Jadavpur University, during the academic year 2017 – 2018. I wish him every success in life.

.....
Prof. (Dr.) Ujjwal Maulik
Head of the Department
Department of Computer Science & Engineering
Jadavpur University, Kolkata – 700032.

.....
Prof. (Dr.) Dipankar Das
Project Supervisor,
Department of Computer Science & Engineering
Jadavpur University, Kolkata – 700032.

.....
Prof. (Dr.) Chiranjib Bhattacharjee
Dean, Faculty council of Engg. & Tech.
Jadavpur University, Kolkata – 700032.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC PROJECT

I hereby declare that this project contains literature survey and original research work by the undersigned candidate, as part of his *MASTER OF COMPUTER APPLICATIONS* studies. All information in this document have been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material results that are not original to this work.

NAME: SUTANU PAUL

ROLL NUMBER: 001510503017

“USER-ITEM BASED HYBRID RECOMMENDATION SYSTEM BY EMPLOYING
MAHOUT FRAMEWORK”

SIGNATURE WITH DATE

**JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY**

CERTIFICATE OF APPROVAL

The foregoing project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR

EVALUATION OF PROJECT:

1. _____

2. _____

(Signature of Examiners)

ACKNOWLEDGEMENT

I express my honest and sincere thanks and humble gratitude to my respected teacher and guide *Prof. (Dr.) Dipankar Das*, Assistant Professor of the Department of Computer Science & Engineering, Jadavpur University, for his exclusive guidance and entire support in completing and producing this project successfully. I am very much indebted to him for the constant encouragement, and continuous inspiration that he has given to me. The above words are only a token of my deep respect towards him for all he has done to take my project to the present shape.

I would like to thank *Mr. Subhabrata Dutta* for valuable support and suggestions to the activities of the project.

Finally, I convey my real sense of gratitude and thankfulness to my family members for being an endless source of optimism and positive thoughts; and last but not the least, my father & mother for their unconditional support, without which I would hardly be capable of producing this huge work.

Sutanu Paul

Examination Roll: MCA186017

Registration No: 133679 of 2015 – 2016

ABSTRACT

Recommender systems have become increasingly popular in recent years, and are utilised in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Recommender systems use Machine Learning techniques and Data mining algorithms to predict what items should suggest to the users based on some previous information related to the users and their relations to the items. It basically offers the users a limited number of products and services which he/she would like to get among the vast amount of all available items. The growth of information on the Internet as well as number of visitors to websites is producing many choices to a customer, but many of those are irrelevant to them. A Recommendation System filters this data and refers the filters data to the users. A Recommendation system is totally based on its training data, the performing algorithms and the recommending approach. There are many popular approaches like user based collaborative filtering, item based collaborative filtering, content based filtering and Hybrid models.

We have implemented three different architectures, item based, user based and factor based hybrid models in order to build recommender system for artist. In order to cope up with the huge amount of data, we have used Apache Mahout on top of Hadoop. Apache Mahout is an open source framework, primarily used for creating scalable machine learning algorithms. Using Mahout we can fasten the process. Mahout offers the coder a ready-to-use framework for doing data mining tasks on large volumes of data.

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Problem Statement	2
1.3. Challenges.....	3
1.4. Contributions	4
2. Related Work	5
3. Dataset Preparation	7
3.1. Data Source.....	7
3.2. Pre-processing of Data	8
4. System Design	12
4.1. Mahout Framework	13
4.2. User Based Recommendation	13
4.3. Item Based Recommendation	14
4.4. Hybrid Recommendation	16
4.4.1. Gender Based Factor Model.....	16
4.4.2. Country Based Factor Model	17
4.4.3. Age Based Factor Model.....	18
4.5. Evaluator Model.....	20
5. Evaluation	21
5.1. Experiment and Results	21
5.2. Comparative Analysis.....	23
5.3. Error Analysis	23
6. Conclusion and Future Work	25
References	26
Appendix A	28
Appendix B.....	38

1. Introduction

The present attempt focuses on developing recommendation systems mostly used in the digital domain. Majority of today's E-Commerce sites like eBay, Amazon, Alibaba etc make use of their proprietary recommendation algorithms in order to better serve the customers with the products they are supposed to like. From business perspective recommendation systems are typically used to enhance the business and increase the sale by helping users discover items they might not have found by themselves and promote sales to potential customers based on their previous selections over the other similar data. A good recommendation system can provide customers with the most relevant products. This is a highly-targeted approach which can generate high conversion rate and make it very effective and smooth to do advertisements. So the problem we are trying to study here is that, how to build effective recommendation systems that can predict products that customers like the most and have the most potential to buy based on the research on some existing models and algorithms like User Based Collaborative Filtering, Item Based Collaborative Filtering and Hybrid Recommendation algorithm which is just combination of other algorithms. They can be used to predict the rating for a product that a customer has never reviewed, based on the data of all other users and their ratings in the system. We implement these three algorithms, and then test them on Last.fm-360k-datasets to do comparisons and generate results. Which have 17,000,000+ ratings and 360,000+ users. Our goal is to build a recommendation system which recommend based on user activity, Item similarity and user profile. We use a Machine learning framework of Apache called Mahout coupled with hadoop to analyse huge amount data. There is another tsv dataset which contains the user profile of all the users like their gender, age, country and their registration date. Such dataset has been further used in hybrid model to improve the performance of the system.

1.1. Motivation

The tendency to use past data for business purpose for each industry and organisation is increasing day by day. Today, around 2.5 Quintillion bytes of data has been created per day and in future more data will be generated. Storing and accessing these data is a separate issue but paper is all about accessing the data. We want to use the bigdata to implement a recommendation system and making it better. There is a urgent demand in the industry for a

recommendation system. The recommendation system is new to the industry and industry want researchers to look over it to make it better. The web sites like fast.fm, netflix, amazon's are gaining billions for their good recommendation system. Netflix have announced 1 million for the recommendation system which can beat their own recommendation system. Actually this is the main reason that people care about recommendation systems because it brings money. Big web giants like Amazon, Netflix, Spotify and etc are investing huge money on recommendation system. It is increase their profit by adding value to the customer. It's saves customer's time and also engages him/her with the website. Apart from that researchers are also interested in it because it's a real Data science problem. It involves Statistics, Machine Learning, and Software engineering.

1.2. Problem Statement

Here, we are going to build a Recommendation System using machine learning algorithms to recommend artists to the last.fm users and predict their ratings. By adding some factor models, we tried to improve the recommended output better. Recommendation system stimulates the sales and increase profits. For our recommendation system, we decided to choose the Last.fm-360k-users dataset which consists of 17,000,000 reviews and 360,000 users. Our recommendation system is not only for rating prediction for users but also it explores the new possibilities. As per our problem, we assume that the users tend to like the products that have a high preference value to the user. We will consider the preference score ranges within 1 to 5. We have implemented several algorithms to predict the scores of each product. The main objective of this project is to develop a music-artist recommendation system. The system will determine the preferences to the artists for the users, based on the analysis of the data generated by their past activity. First, we build some traditional systems and then in hybrid model, we combine their results. Also, it has been taken into account that we do not always want to hear the same artists or genres or favourite bands. Sometimes, a listener needs to be surprised to enjoy a new discovery. In order to implement that, we use some factor models with the hybrid system. The factor models are implemented to refer them the artists which are generated based on traditional approaches but their predicted preference value is not only based on the traditional system but also it depend on their gender, age and location.

Now, let's take an simple example to understand how does a recommender system recommend.

Table -1: A simple example to understand recommender system

DATASET :			RESULT :
UserId	ArtistId	Rating	
1,	00,	1.0	Recommendation result for User 2 : RecommendedItem [item:3, value:4.5] RecommendedItem [item:4, value:4.0]
1,	01,	2.0	
1,	02,	5.0	
1,	03,	5.0	
1,	04,	5.0	
2,	00,	1.0	
2,	01,	2.0	
2,	05,	5.0	
2,	06,	4.5	
2,	02,	5.0	
3,	01,	2.5	
3,	02,	5.0	
3,	03,	4.0	
3,	04,	3.0	
4,	00,	5.0	
4,	01,	5.0	
4,	02,	5.0	
4,	03,	0.0	

1.3. Challenges

1. The last.fm-360k-dataset is a large dataset containing 17 million ratings and 360k users. To handle that huge data, we use hadoop's Map Reduce framework.
2. The last.fm-360k-dataset contains user id and song as string attribute and number of plays as integer value with no range. Since, the data model for Mahout's recommender has a specific format like long data type for user id and item id and preference point as double. To resolve this situation, we use hash mapping with each

string user id and item id with a unique number and we normalize the number of plays within a range.

3. Producing improved recommendation and handling many recommendations efficiently is also a challenge but Mahout's ready to use framework solve this case.

To address these issues, we have explored several collaborative filtering techniques such as the user based approaches, item based approaches. Item based approach identifies the relationship between items and indirectly computes recommendations for users based on these relationships. The user based approach was also studied; it identifies relationships between users with similar taste and computes recommendations based on these relationships. The paper analyzes algorithms of user based and item based collaborative filtering techniques for recommendation generation. Moreover, a simple experiment was conducted using a Machine Learning framework of Mahout. We conclude by proposing our approach that might enhance the quality of recommender systems.

1.4. Contributions

Recommendation systems are evolving with the time and getting better and better. Applying approaches like user based CF, item based CF and user profile based model to build a weighted hybrid recommendation system. Machine Learning algorithms of Mahout, a framework of a hadoop has been used. This project is a combined application of machine learning over Big Data. Also, it explores the Mahout framework whereas the hybrid system copes up with some other criteria based on user's profile. In this project we have analysed the two traditional models and one hybrid model in brief.

2. Related Work

There are two main approaches for a recommendation system has been used. These are most popular algorithms, i.e., the content-based filtering approach and the collaborative filtering approach. Content-based filtering approach Based on the item profiles and user profile where the relation between the user profile and item profiles are registered at the beginning. Based on item profile the system finds the similar items and recommends only those similar items that are highly relevant to the similar user profiles. Examples of such systems are NewsWeeder [5], Infofinder [6], and News Dude [7].

On the other hand the collaborative filtering have two subtypes one is user based CF other is item based CF these algorithms cluster the data according to the user based or item based approach and group them as based on similarity measurement and recommend them accordingly. This type of recommendation system is mentioned by surendra[8] and Hintone [9]. Instead of computing the similarities between the item profiles and the user profiles the collaborative approach computes the past experience of the user over the item. As described above the purpose of the collaborative approach is to make recommendation among the users in the same group, the recommendation of the Collaborative method depends on the users' interests and the interests are derived from the users' access histories.

Some websites even tracks the item web pages searched or viewed by the user and those data taken account to derive preference value. Therefore, the users will never get a recommendation of music objects belonging to the music groups they never accessed before as per CB method. That is, the Collaborative method tends to provide expected and interesting music objects for users. The purpose of the CF method is to provide unexpected findings due to the information sharing between relevant users. To refer to the information from other users, we group the users first. There are the technique proposed in [10] for user grouping. Examples of such systems are Ringo [11] and Siteseer [12]. In the collaborative filtering approach, the system may have a high possibility to recommend unexpected data items by the nature of information sharing.

Some systems use both contend-based and collaborative filtering approaches which exploit both ratings and content information. Like recommendation system discussed in [4] which depart from the traditional social-filtering approach by framing the problem as one of classification, rather than artifact rating. However, it differs from content based filtering methods in that social information, i.e. other users' ratings, will be used in the inductive learning process. For example, Tapestry [13] and GroupLens [14] allow users to comment on

Netnews and group users by computing the similarities of their ratings of newsgroups. In addition, for the process of recommendation, users have to specify their profiles and describe the features of data items which they are interested in. The user interests are derived from the types, the actors, and the scenarios of items that the user accessed in the past. The users are also required to specify the satisfactory degrees of the accessed items. With respect to items, users who specify similar satisfactory degrees will be grouped together for collaborative recommendation.

Similarly, the Personalized Television system [15] provides a personalized list of recommended programs. The FAB system [3] analyzes the accessed web pages to derive the user profiles and compares the user profiles to group users for collaborative recommendation.

Now Mahout is an open source machine learning library from Apache. Using Mahout we can combine the traditional algorithms. Mahout is written in Java and primarily focused on recommendation engines, clustering, and classification. Its core algorithms are implemented on top of Hadoop so that it can scale on large datasets [20].

3. Dataset Preparation

The recommendation websites stores the data generated per day. It stores the details of the user like their age, gender, country etc and along with the data related to user' relationship with content .The websites usually stores these data which is supposed to be used for recommendation.

3.1. Data Source

Last.fm is a music website, founded in the United Kingdom in 2002. Using a music recommender system called "Audioscrobbler", invented by Richard Jones. Last.fm builds a detailed profile of each user's musical taste by recording details of the tracks the user listens to, either from Internet radio stations, or the user's computer or many portable music devices. This information is transferred ("scrobbled") to Last.fm's database either via the music player itself (including, among others Spotify, Deezer, Tidal, and MusicBee) or via a plug-in installed into the user's music player. The data is then displayed on the user's profile page and compiled to create reference pages for individual artists as er wiki description [21]. I have chosen last.fm-360k-dataset. I have downloaded the data set contained in a tar file from Last.fm Dataset - UPF[22]. The detailed information about the tar file is given below:

. Files:

usersha1-artmbid-artname-plays.tsv (MD5: be672526eb7c69495c27ad27803148f1)
 usersha1-profile.tsv (MD5: 51159d4edf6a92cb96f87768aa2be678)
 mbox_sha1sum.py (MD5: feb3485eace85f3ba62e324839e6ab39)

. Data Statistics:

File usersha1-artmbid-artname-plays.tsv:

Total Lines:	17,559,530
Unique Users:	359,347
Artists with <u>MBID</u> :	186,642
Artists without <u>MBID</u> :	107,373

. Data Format:

The data is formatted one entry per line as follows (tab separated "\t"):

File `usersha1-artmbid-artname-plays.tsv`:

```
user-mboxsha1 \t musicbrainz-artist-id \t artist-name \t plays
```

File `usersha1-profile.tsv`:

```
user-mboxsha1 \t gender (m|f|empty) \t age (int|empty) \t country (str|empty) \t signup
(date|empty)
```

. Example:

`usersha1-artmbid-artname-plays.tsv`:

```
000063d3fe1cf2ba248b9e3c3f0334845a27a6be \t a3cb23fc-acd3-4ce0-8f36-
1e5aa6a18432 \t u2 \t 31
```

...

`usersha1-profile.tsv`

```
000063d3fe1cf2ba248b9e3c3f0334845a27a6be \t m \t 19 \t Mexico \t Apr 28, 2008
```

...

There is two main files ‘`usersha1-artmbid-artname-plays.tsv`’ and ‘`usersha1-profile.tsv`’. As we know from the above mentioned information that ‘`usersha1-artmbid-artname-plays.tsv`’ is the main file which contains the main file which has the data based on previous user-Item relation i.e. users past experiences. The other one is the data set based on user’s profile. Our final goal is to build a hybrid recommendation system which performs better than the traditional systems. CF recommender does not require the user profiles. It only needs the user-Item relationships to compute the recommendation algorithm.

3.2. Pre-processing of Data

Now `usersha1-artmbid-artname-plays.tsv` dataset contains four tab separated columns as mentioned above:

```
“user-mboxsha1 \t musicbrainz-artist-id \t artist-name \t plays”
```

We pre-process the data in the following steps.

1. The ‘`usersha1-artmbid-artname-plays.tsv`’ data set contains user id, artist id, artist name or plays. There may be some incomplete information like any of user id, artist id, artist name or

plays is missing. We simply discard those rating information and consider the remaining. So, if all the four attributes are not available then we will simply discard that rating by applying a filter condition over the data set we do that.

2. Preference point is a number which gives a value to the item for a user that how much he/she likes it. Since we have any preference value but the number of plays of an artist is available. For a user if number of hits or plays for a song is higher than that of the other song then its preference point will be high. So, we can define a preference value which is based on a user's number of plays but every user's liking should be rate in a same range of interval. Because some user listen music frequently and some user listen music often. Let us take an example :- a user listen music frequently but listen a particular song a certain number of times and some other user listen that song same number of times but this user is less interested in music so he usually less frequent to the website. But does not mean that the 1st user like the song same the 2nd user. Because 2nd users' listening percentage to that song is more than that of 1st user so, we can assume that the 2nd user likes the song more than first singer. So, we can solve this problem by normalize the number of plays and derive a preference value out of it which ranges in an interval say 1 to 5 per user which will be based on the listening percentage. I.e. the artist he/she listens most is the item with the highest rating say 5 and the artist he/she listens least is the item with lowest rating. I.e. if the number of plays is high then that item is a highly preferable item for the user and if it is less listened artist then it is less preferable to the user. Let **a** and **b** are the number of highest and lowest number plays respectively of a user among his/her all past listened artists. Now suppose **x** is the number of plays for any random artist then the rating generating function for that artist will be:

$$\text{Rate}(x) = 1 + \frac{x-a}{b-a} * 4$$

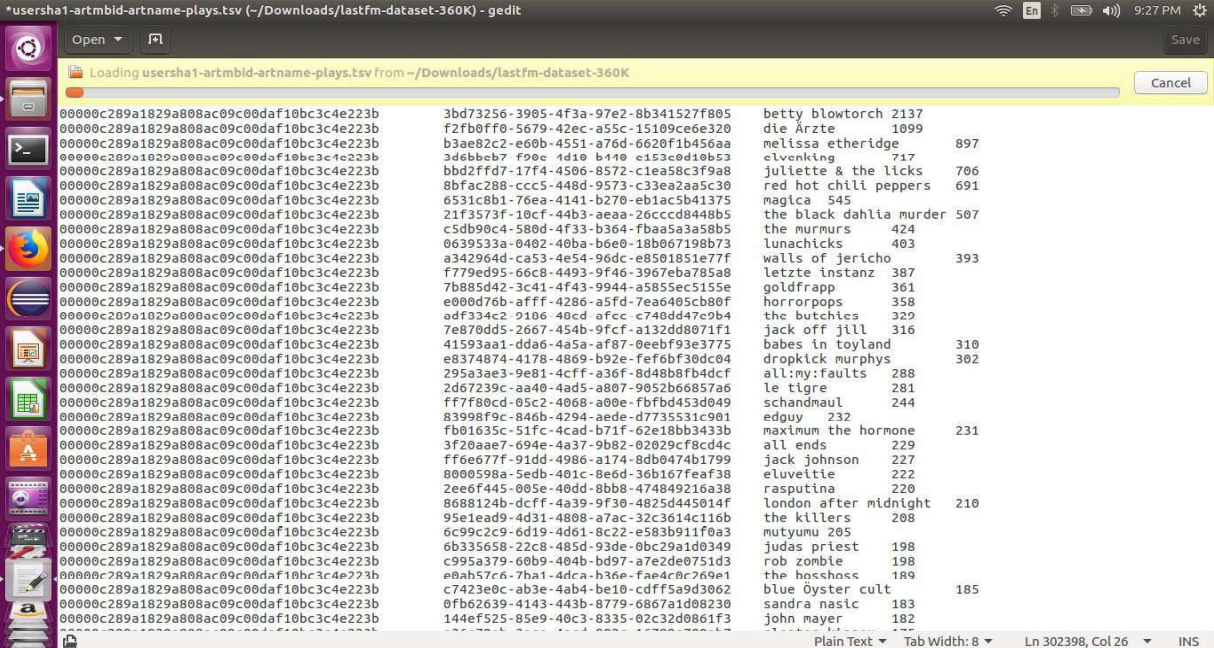
This value ranges from 1 to 5.

3. The user-mboxsha1 is a unique identifier of users, is a 40 character string, musicbrainz-artist-id is a unique identifier for Artists, is a 36 character string, artist-name is a character string and number of plays is an integer which count the number of times the artist is hit by the user. But this data set should be model in a meaningful format which can be identified by the Mahout framework. As per Mahout framework the data model for input data should be a csv file whose first attribute should be user id and data type must be long and the second attribute is item-id and this is also a long data type and third data type is a double data type

which is called the preference id and all should be formatted in comma separated format like example-1 dataset. So, we need to process the data in such a way that 17,000,000 ratings converted to the csv format. To do that I have used hash map and which maps the string to a long data type and compute preference value from the number of plays and keep the mapping until the recommendation is done. I.e. unique integers for per user-mboxsha1 and musicbrainz-artist-id will be generated and a new data set will be formed with those new integer ids. With the new data set recommendation will be done and end of the recommendation, in the result set it maps back the String id's or artist name by using the hash map. We built a StringToInt class to handle this mapping.

The data set is huge and tough to handle it we use hadoop's Map Reduce technique. I have used a hadoop's single node cluster if we can increase the number of node in future then we can test the recommending time is being reduced or not and if reduce then by how much. This mapper takes the first attribute i.e. the user id as the key and the remaining as value and at reducer phase it compute the necessary values for each key.

After all the pre-processing part the data set transformed in to a new dataset in hadoop's output file named as 'part-00000'. We store that file in a separate space and the recommendation will done on this file. It's a csv file with three attributes and comma separated. This is the specific file format for Mahout's recommendation system. I have provided screen of the data sets before pre-processing [fig-1] and after pre-processing [fig-2].



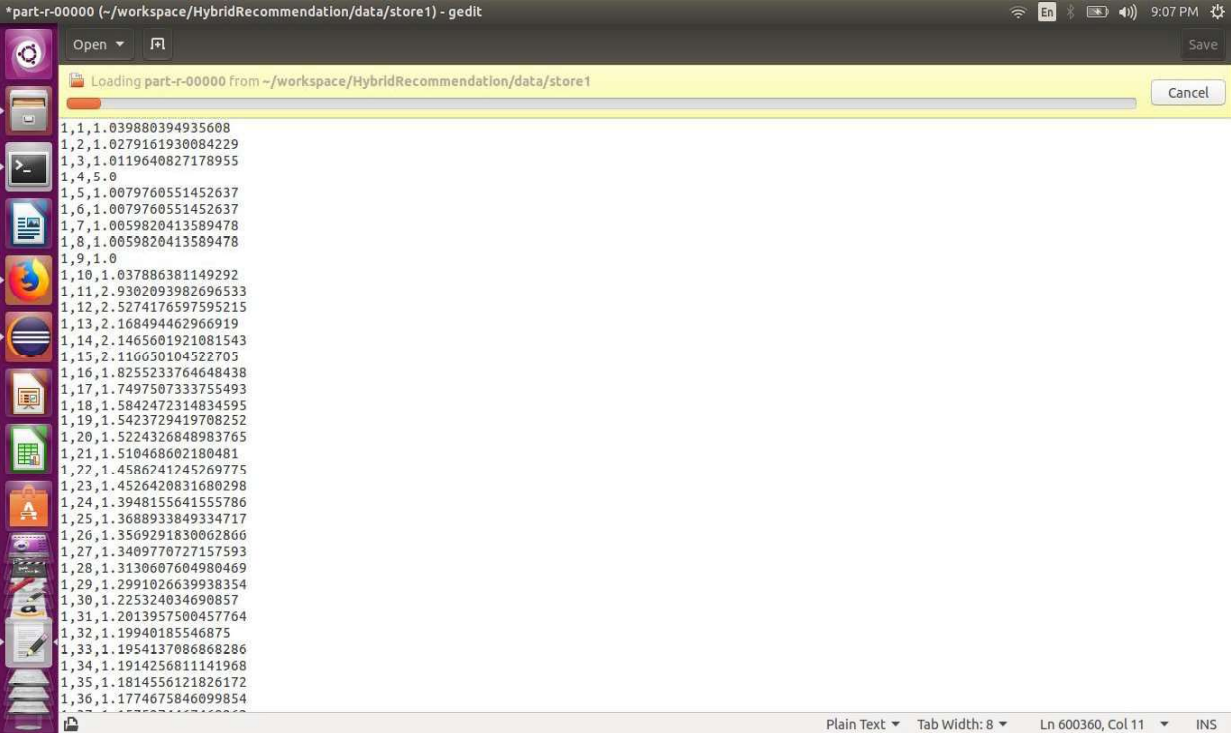
The screenshot shows a text editor window titled '*usersha1-artmbid-artname-plays.tsv' with the following content:

```

00000c289a1829a808ac09c00daf10bc3c4e223b 3bd73256-3905-4f3a-97e2-8b341527f805 betty blowtorch 2137
00000c289a1829a808ac09c00daf10bc3c4e223b f2fb0ff0-5679-42ec-a55c-15109ce6e320 die Ärzte 1099
00000c289a1829a808ac09c00daf10bc3c4e223b b3ae82c2-e60b-4551-a76d-6620f1b456aa melissa etheridge 897
00000c289a1829a808ac09c00daf10bc3c4e223b 3d6bbeb7-f90e-4d10-b110-e153c0d10b53 clvenking 717
00000c289a1829a808ac09c00daf10bc3c4e223b bbd2ffd7-17f4-4506-8572-c1ea58c3f9a8 juliette & the licks 706
00000c289a1829a808ac09c00daf10bc3c4e223b 8bfac288-ccc5-448d-9573-c33ea2aa5c30 red hot chili peppers 691
00000c289a1829a808ac09c00daf10bc3c4e223b 6531c8b1-76ea-4141-b270-eb1ac5b41375 magica 545
00000c289a1829a808ac09c00daf10bc3c4e223b 21f3573f-10cf-44b3-aeaa-26ccc0d440b5 the black dahlia murder 507
00000c289a1829a808ac09c00daf10bc3c4e223b c5d909c4-580d-4f33-b364-fbaa5335b5b5 the murmurs 424
00000c289a1829a808ac09c00daf10bc3c4e223b 0639533a-0402-40ba-b6e0-18b067198b73 lunachicks 403
00000c289a1829a808ac09c00daf10bc3c4e223b a342964d-ca53-4e54-96dc-e8501851e77f walls of jericho 393
00000c289a1829a808ac09c00daf10bc3c4e223b f779ed95-66c8-4493-9f46-3967eba785a8 letzte instanz 387
00000c289a1829a808ac09c00daf10bc3c4e223b 7b885d42-3c41-4f43-9944-a5855ec5155e goldfrapp 361
00000c289a1829a808ac09c00daf10bc3c4e223b e000d76b-afff-4286-a5fd-7ea6405cb80f horrorpops 358
00000c289a1829a808ac09c00daf10bc3c4e223b adf334c2-9106-40cd-afcc-c740dd47e9b4 the butchies 329
00000c289a1829a808ac09c00daf10bc3c4e223b 7e870d5-2667-454b-9fcf-a132dd8071f1 jack off jill 316
00000c289a1829a808ac09c00daf10bc3c4e223b 41593aa1-dda6-4a5a-af87-0eebf93e3775 babes in toyland 310
00000c289a1829a808ac09c00daf10bc3c4e223b e8374874-4178-4869-b92e-fe6fb30dc04 dropkick murphys 302
00000c289a1829a808ac09c00daf10bc3c4e223b 2953aa3e-9eb1-4cff-b36f-8d48b8f04dcf all-my-faults 288
00000c289a1829a808ac09c00daf10bc3c4e223b 2d67239c-aa40-4ad5-a807-9852b66857a6 le tigre 281
00000c289a1829a808ac09c00daf10bc3c4e223b ff7f80cd-05c2-4068-a00e-fbfbd453d049 schandnau 244
00000c289a1829a808ac09c00daf10bc3c4e223b 83998f9c-846b-4294-aede-d7735531c901 edguy 232
00000c289a1829a808ac09c00daf10bc3c4e223b fb01635c-51fc-4cad-b71f-62e18bb3433b maximum the hormone 231
00000c289a1829a808ac09c00daf10bc3c4e223b 3f20aae7-694e-4a37-9b82-02029cf8cd4c all ends 229
00000c289a1829a808ac09c00daf10bc3c4e223b ff6e677f-91dd-4986-a174-8db0474b1799 jack johnson 227
00000c289a1829a808ac09c00daf10bc3c4e223b 8000598a-5edb-401c-8e6d-36b167feaf38 eluveitie 222
00000c289a1829a808ac09c00daf10bc3c4e223b 2ee6f445-005e-40dd-8bb8-474849216a38 rasputina 220
00000c289a1829a808ac09c00daf10bc3c4e223b 8688124b-dc1ff-4e39-9f30-4825d445014f london after midnight 210
00000c289a1829a808ac09c00daf10bc3c4e223b 95e1ead9-4d31-4808-a7ac-32c3614c116b the killers 208
00000c289a1829a808ac09c00daf10bc3c4e223b 6c99c2c9-6d19-4d61-8c22-e583b911f0a3 mutyumu 205
00000c289a1829a808ac09c00daf10bc3c4e223b 6b335658-22c8-485d-93de-0bc29a1d0349 judas priest 198
00000c289a1829a808ac09c00daf10bc3c4e223b c995a379-60b9-404b-bd97-a7e2de0751d3 rob zombie 198
00000c289a1829a808ac09c00daf10bc3c4e223b e0ab57c6-7ha1-4dca-b36e-fae4c0c769e1 the hoxhoss 189
00000c289a1829a808ac09c00daf10bc3c4e223b c7423e0c-ab3e-4ab4-be10-cdff5a9d3062 blue oyster cult 185
00000c289a1829a808ac09c00daf10bc3c4e223b 0fb2639-4143-443b-8779-6867a1d08230 sandra nasic 183
00000c289a1829a808ac09c00daf10bc3c4e223b 144ef525-85e9-40c3-8335-02c32d0861f3 john mayer 182

```

Raw Dataset [fig-1]



```
*part-r-00000 (~workspace/HybridRecommendation/data/store1) - gedit
Loading part-r-00000 from ~/workspace/HybridRecommendation/data/store1
Cancel
1,1,1.039880394935608
1,2,1.0279161930084229
1,3,1.0119640827178955
1,4,5.0
1,5,1.0079760551452637
1,6,1.0079760551452637
1,7,1.0059820413589478
1,8,1.0059820413589478
1,9,1.0
1,10,1.037886381149292
1,11,2.9302093982696533
1,12,2.5274176597595215
1,13,2.168494462966919
1,14,2.1465601921081543
1,15,2.116650104522703
1,16,1.8255233764648438
1,17,1.7497507333755493
1,18,1.5842472314834595
1,19,1.5423729419708252
1,20,1.5224326848983765
1,21,1.510468602180481
1,22,1.4586741245269775
1,23,1.4526420831680298
1,24,1.3948155641555786
1,25,1.3688933849334717
1,26,1.3569291830062866
1,27,1.3409770727157593
1,28,1.3130607604980469
1,29,1.2991026639938354
1,30,1.225324034690857
1,31,1.2013957500457764
1,32,1.19940185546875
1,33,1.1954137086868286
1,34,1.1914256811141968
1,35,1.1814556121826172
1,36,1.1774675846099854
Plain Text Tab Width: 8 Ln 600360, Col 11 INS
```

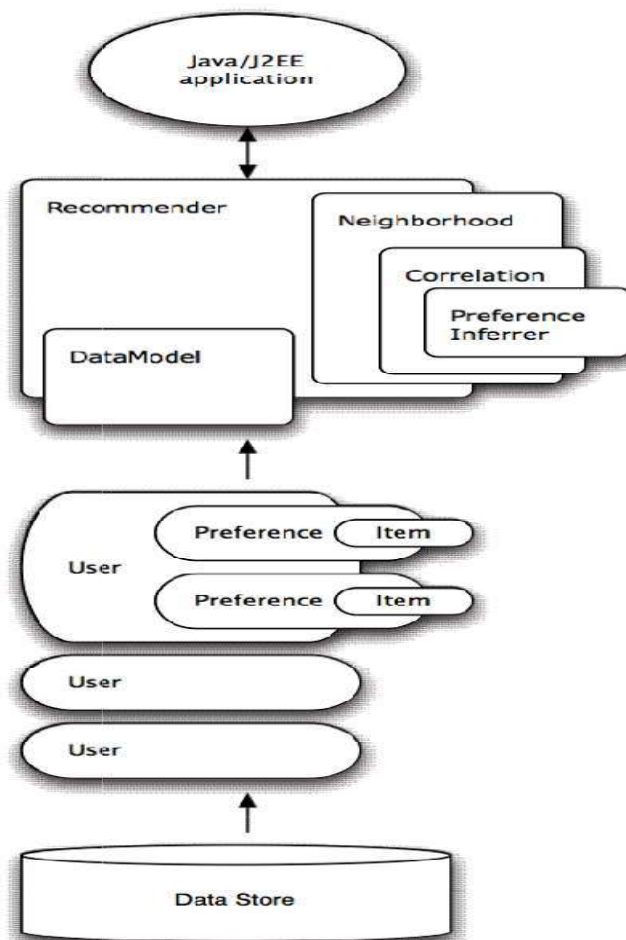
Pre-processed Dataset [fig-2]

4. System Design

Let us understand a user-item rating matrix in order to understand CF Recommender systems. It is an $n \times m$ matrix where n users rated m items. Here, each of the cells corresponds to the rating given to item 'i' by the user 'x'. i.e each row represent the rating vector corresponding to a user. This user rating matrix is typically sparse, as most users do not rate most of the items. Thus, the goal of the present system is to recommend those unrated cells .i.e. those items which are not reviewed by the user. The user under current consideration is referred as the *active user*. The three models are given below :

1. User Based Collaborative Filtering (CF)
2. Item Based Collaborative Filtering (CF)
3. Hybrid approaches: These methods combine both collaborative based approaches along with the factor models based on user profiles.

The basic architectural design of the recommendation systems implemented in Mahout [1] is given in [fig-3].



Mahout Architecture [fig-3]

4.1. Mahout Framework

Apache Mahout is a distributed linear algebra framework implemented on java and coupled with hadoop to boost up machine learning algorithms. It focused primarily on the areas of collaborative filtering, clustering and classification. Many of the implementations use the Apache hadoop platform. Mahout also provides Java libraries for common maths operations (focused on linear algebra and statistics) and primitive Java collections. Mahout is a work in progress. The number of implemented algorithms has grown quickly, but various algorithms are still missing [21]. Still, it is a powerful tool in research field to deal with machine learning. Mahout is providing in-built ready to use basic machine learning algorithms.

Weakness of Mahout includes poor visualization and less support for scientific libraries compared to Spark for Python and what Python has inherited from R. Mahout has a lot of dependencies which can be a drag if one is simply using it for Spark jobs.

4.2. User Based Recommendation

User based recommendation is a type of collaborative filtering where user-based algorithm produces recommendation list for the user according to the view of other users with similar taste. Underlying concept is, if the ratings of some items rated by some users are similar, the rating of other items rated by these users will also be similar. Refer to the previous example-1 we make table-2.

Table-2 : User based approach

	ItemID (ArtistID)						
UserID	00	01	02	03	04	05	06
1	1.0	2.0	5.0	5.0	5.0	NA	NA
2	1.0	2.0	5.0	NA	NA	5.0	4.5
3	NA	2.5	5.0	4.0	3.0	NA	NA
4	5.0	5.0	5.0	0.0	NA	NA	NA

Now let us look at the ratings among user-1, user-2 and user-3. Since for most of the cases their ratings are similar, we can put them in a same group. If we want to refer some item to user-2, then it must be item-03 and item-04 because of high rating from user-1 and user-3.

There are some metric functions in Mahout which measures the similarity measures between the users and based on that value it group the users. Based on a threshold value it will allow other user to be in same neighbourhood. Collaborative Filtering algorithm that uses a threshold value to determine the neighbourhood is discussed here. The algorithm can be summarized in the following steps:

- Find the similarity value between the users. This similarity metric is Pearson correlation similarity according to our problem.

$$S(x, y) = \frac{\sum_{i=1}^n (r(x_i) - \overline{r(x)})(r(y_i) - \overline{r(y)})}{\sqrt{\sum_{i=1}^n (r(x_i) - \overline{r(x)})^2 \sum_{i=1}^n (r(y_i) - \overline{r(y)})^2}} \quad ..(1)$$

- Select n active users that have the highest similarity up to a threshold value.
- Find the items which are not reviewed by the user x yet and reviewed by most of other users among n users.
- Compute a predicted rating, from the top similarity. Let x is the current user, y is any of the similar users and i is the item then formula for prediction for user x over item i will be:

$$P(x, i) = \overline{r(x)} + \frac{\sum_{y=1}^n (r(y_i) - \overline{r(y)}) * S(x, y)}{\sum_{y=1}^n S(x, y)} \quad ..(2)$$

Where $\overline{r(x)}$ and $\overline{r(y)}$ is the mean rating of user x and user y respectively.

$r(x_i)$ and $r(y_i)$ are rating of user y and user x over item i respectively.

$S(x, y)$ is the similarity between user x and user y .

Since, Mahout has in-built ready to use functions and classes for recommendation. So, we don't need to work hard for coding part. The recommendation system is implemented in Java and computes 20 recommendations for particular user.

4.3. Item Based Recommendation

Item based recommendation is another type of collaborative filtering. The idea is the ratings of some items rated by some users are similar. The rating by the other users to those items will also be similar, i.e. if some user rates some items with similar rating then these

items will be similar kind of item. We again refer to the previous example-1 and make the user item rating matrix table (table-3) to understand the concept.

Table-3 : Item based approach

	ItemID (ArtistID)						
UserID	00	01	02	03	04	05	06
1	1.0	2.0	5.0	5.0	5.0	NA	NA
2	1.0	2.0	5.0	NA	NA	5.0	4.5
3	NA	2.5	5.0	4.0	3.0	NA	NA
4	5.0	5.0	5.0	0.0	NA	NA	NA

Now let us look at the ratings among item-02, item-03 and item-04. Since for most of the cases their ratings are similar, we can put them in a same group. If we want to refer to some item user-2 then it must be item-03 and item-04 because they are in a same group with item-02.

The metric functions in Mahout is used to measure the similarity measures between the items and based on the rating by the user. Collaborative Filtering does not need a threshold value for neighbourhood. The algorithm can be summarized in the following steps:

- Find the similarity value between the items. This similarity metric is Pearson correlation similarity according to our problem.

$$S(i, j) = \frac{\sum_{x=1}^n (r(x_i) - \overline{r(x)}) (r(x_j) - \overline{r(x)})}{\sqrt{\sum_{x=1}^n (r(x_i) - \overline{r(x)})^2 \sum_{x=1}^n (r(x_j) - \overline{r(x)})^2}} \quad ..(3)$$

- Select active users that have the highest similarity.
- Find the items which are not reviewed by the user x yet and reviewed by most of other users among n users.
- Compute a prediction, from the top similarity. Let x is the user and i is the item then formula for prediction for user x over item i will be:

$$P(x, i) = \overline{r(i)} + \frac{\sum_{j=1}^n (r(x_j) - \overline{r(x)}) * S(i, j)}{\sum_{j=1}^n S(i, j)} \quad ..(4)$$

Where $\overline{r(x)}$ and $\overline{r(y)}$ is the mean rating of user x and user y respectively.
 $r(x_i)$ and $r(y_i)$ is rating of user y and user x over item i respectively.

$S(i,j)$ is the similarity between item i and item j .

Now, we can proceed to our main goal, the hybrid model. This model combines both the recommendations above and user profile to recommend better.

4.4. Hybrid Recommendation

This Hybrid recommendation system is a combination of the recommendation systems mentioned above and also it associates the measurement of the user profile data. There are 7 types of hybrid recommendation systems: weighted, switching, mixed, feature combination, feature augmentation, cascade, and meta-level.

This recommendation system is a type of mix-weighted hybrid recommendation together with the factor models based on user profile data. We combine some the recommendation from user based CF and some from item based CF and combine them and then we put them in a single list. Then divide their Preference value in some parts.

- P_1 : 70% preference value based on their collaborative algorithms.
- P_2 : 10% preference value based on their gender based model.
- P_3 : 10% preference value will based on their age
- P_4 : 10% preference value will be based on their location.

Now after data cleaning all the parts are same but after that we combine all the approaches on the pre-processed dataset like previous examples we find the similarity between two users or items based on that similarity.

4.4.1. Gender Based Factor Model

To optimize our recommendation system we use user profile data; underlying concept being, if an artist is famous among a particular gender then it is obvious that the artist is highly preferable to that gender. For example Justin Bieber is famous among girls or ladies then he is highly preferable to a female user.

Let, i is an artist and u_i are the users who have already reviewed i . Let x is an active user then according to x , the preference value of the artist i will be say $P_2(x, i)$. Let total male count is M and total female count is F . and the song is preferred by f female user and m male users.

Then female percentage is $\frac{f*100}{F}$ and male percentage is $\frac{m*100}{M}$ who performed the artist.

Case 1: If the user is female, then her preference will be high if $\frac{\frac{f*100}{F}}{\frac{m*100}{M}} = \frac{f*M}{m*F} > 1$ and

Preference will be low if $\frac{f*M}{m*F} < 1$

We call it advantage ratio over gender, $r = \frac{f*M}{m*F}$.

Case 2: Similarly, for a male user it will be $r = \frac{m*F}{f*M}$.

Then for our recommendation system the preference value should be in between $[0,1]$.

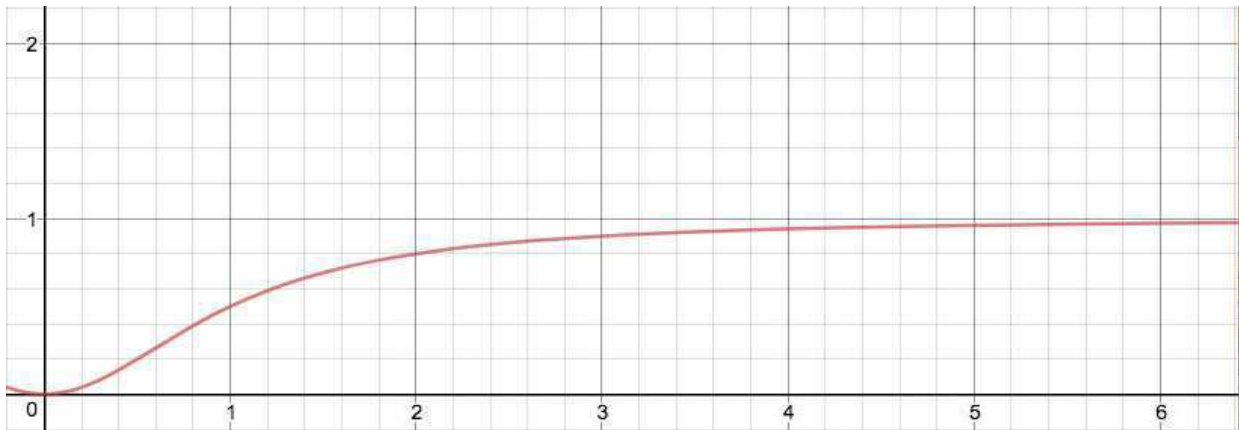
So, we define a function which maps r in a range $[0,1]$.

Let preference value based on gender will be determined by the following function.

$$P_2(\mathbf{x}, \mathbf{i}) = \frac{r^2}{1+r^2}$$

$P_2 : [0, \infty) \rightarrow [0, 1]$, It's a bijective mapping

The value lies between 0 to 1 and for a neutral case, it will give 0.5 [fig-6].



P₂ Graph [fig-4]

4.4.2. Country Based Factor Model

In our recommendation system we use user's location data. Here the concept is if a artist is famous in some country then it is much likely that an user belonging to that country would prefer that artist. For example, being famous in India, Arijit Singh is most likely to be preferred by an Indian user.

Let, \mathbf{i} is an artist and \mathbf{u}_i are the users who have already reviewed \mathbf{u}_i . Let \mathbf{x} is an active user then according to \mathbf{x} the preference value of the artist \mathbf{i} will be say $P_3(\mathbf{x}, \mathbf{i})$. Let count of users from

all countries is T and count of users from user x 's country among all users is C . Count of users among u_i is t and count of users among u_i is c .

Then percentage users from the country of the user x among all users is $\frac{C*100}{T}$ and

Percentage users from the country of the user x among u_a is $\frac{c*100}{t}$. Then,

Her preference will be high if $\frac{\frac{c*100}{t}}{\frac{C*100}{T}} = \frac{c*T}{t*C} > 1$ and

Preference will be low if $\frac{c*T}{t*C} < 1$

We call it advantage ratio over country, $r = \frac{c*T}{t*C}$

Then for our recommendation system the preference value based on country will be determined by the following function.

$$P_2(\mathbf{x}, \mathbf{i}) = \frac{r^2}{1+r^2}$$

This value is lie between 0 to 1 an for a neutral case it will give 0.5 [fig-6]

4.4.3. Age Based Factor Model

The concept behind incorporating user's age in our recommendation system is, if an artist is famous among a certain age group then that artist is highly preferable to a user from that age group. So, if the variance of difference between the ages of the users who prefer that item and the current user is smaller, then the artist is highly preferred by the person.

Let, \mathbf{i} is an artist and u_i are the users who have already reviewed and let their age is l_u . Let \mathbf{x} is an active user then according to \mathbf{x} the the preference value of the artist \mathbf{i} will be say $P_4(\mathbf{x}, \mathbf{a})$. let \mathbf{x} 's age is l ,

$$D = \sqrt{\frac{1}{n-1} \sum_{u=0}^n (l_u - l)^2}$$

$$\bar{l} = \frac{\sum_{u=1}^n l_u}{n}, \bar{l} \text{ is the mean}$$

Then the coefficient of variant say, $v = \frac{D}{\bar{l}}$.

v lies between 0 to 1 and it is minimum when variation of ages is less i.e. for that case it's a highly preferable item. On the other hand if v is near to 1 then this is less preferable item. Then for our recommendation system the preference value based on country will be determined by the following function:

$$P_3(\mathbf{x}, \mathbf{i}) = 1 - v.$$

This value also lies between 0 to 1.

The main algorithm can be summarized in the following steps which combines all the functions together:

- Find the similarity value between the users and similarity between items using the similarity metric mentioned above (1) and (3).
- Using the traditional recommenders (user based CB and user based CF) compute the recommend item and merge them.
- Compute a predicted using equation (2) and (3) respectively say it's R. Preference will be depend on this rating then P1 will be 40% of R.
i.e. $P_1 = R * 0.7$
- Using the gender based factor model find the preference value based on user's gender say it's P₂.
- Using the country based factor model find the preference value based on user's country say it's P₃.
- Using the age based factor model find the preference value based on user's age say it's P₄.
- Now calculate the new predicted Rating of user **X** over item **i**.

$$R(\mathbf{x}, \mathbf{a}) = W_1 * P_1 + W_2 * P_2 + W_3 * P_3 + W_4 * P_4$$

Where W_1, W_2, W_3 and W_4 are the weightage.

$$W_1 = 3.5, W_2 = 0.5, W_3 = 0.5 \text{ and } W_4 = 0.5$$

$$0 \leq P_1, P_2, P_3, P_4 \leq 1$$

$$\text{So } 0 \leq R(\mathbf{x}, \mathbf{a}) \leq 5.$$

- Now calculate the new predicted rating of user **x** over item **i**. Sort it based on this predicted rating and recommend them best among them.

We have also evaluated the recommendation system above. For that we have created another evaluator class. Mahout provide us these various features to create, implement and test the results so, it reduces the effort and complex algorithms to implement. Because all in there in Mahout.

4.5. Evaluator Model

Mahout provide us various evaluators which we can employ upon our recommendation systems and verify the results. We use IRStatisticsEvaluator to evaluate our Recommendation system it takes a small percentage for training and recommend based on that and calculates that value is how much correct. To evaluate that, we are going to measure the precision, recall and f1 score.

Where ,

$$\text{precision} = \frac{|{\text{relevant documents}} \cap {\text{retrieved document}}|}{|{\text{retrieved document}}|}$$

$$\text{recall} = \frac{|{\text{relevant documents}} \cap {\text{retrieved document}}|}{|{\text{relevant documents}}|}$$

$$\text{f1 score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Here mahout takes a percentage of the data model for training purpose and fill the remaining cells of the user-item rating matrix using recommending algorithm. Now, it finds the some difference measurement of the actual ratings and recommended ratings. Then the evaluator model calculates these above mentioned parameters according to their definition.

5. Evaluation

The experiments regarding the recommendation systems, their results and analysis on their result set are discussed in this section.

5.1. Experiment and Results

We have run our three recommendation systems over the 17,000,000 ratings and we observe the different recommendations recommended by these three recommendation engines for a particular user. The output consists 20 artist and their predicted ratings provided by the different recommendation systems.

The user id is: 00000c289a1829a808ac09c00daf10bc3c4e223b

Table-4: Recommended result for an user by all three recommender systems

User based Recommender		Item Based Recommender		Hybrid Recommender	
Artist	Ratings	Artist	ratings	Artist	ratings
messiah j & the expert	5.0	the stone roses	5.0	richard Elliot	4.7121
ion storm	4.8691	deep insight	5.0	Jazzmasters	4.6394
steinar albrigtsen	4.6045	der plan	5.0	Cartola	4.5940
joaquin phoenix & reese witherspoon	4.5824	bill evans	5.0	ana cañas	4.5798
d.batistatos	4.4402	fefe Dobson	5.0	Gojira	4.5726
dumonde aka deleon and jamx	4.4293	Silverstein	5.0	Acidman	4.5530
femme fatale	4.3571	r.e.m.	5.0	louis prima	4.5449
kerbenok	4.2243	Clawfinger	5.0	Avril	4.5420

the apostles	4.1774	the go! Team	5.0	Am	4.5318
dab (digital analog band)	4.1538	laura branigan	5.0	entombed	4.5313
ross parsley	4.1098	marissa nadler	5.0	the adolescents	4.5003
david rovics	4.1079	why?	5.0	Nico	4.4919
druhá tráva	4.0957	Brujeria	5.0	Paulinho da viola	4.4850
shael riley	4.0909	lars winnerbäck	5.0	everette harp	4.4809
static revenger	4.0565	amanda rogers	5.0	arvo part	4.4778
schola hungarica	4.0360	Dredg	5.0	lee ryan	4.4776
Tanzmuzik	4.0294	pink Floyd	5.0	dire straits	4.4668
yellow blackbird	3.9802	white denim	5.0	Inmigrantes	4.4665
tom novy feat. abigail bailey	3.9686	Atmosphere	5.0	junior senior	4.4508
woods of ypres	3.9466	Festival	5.0	the paul butterfield blues band	4.4410

Here we can see that the rating of item based system and user based system is comparatively higher than the hybrid system. But this does not mean that hybrid system works poor. Because the rating is a predicted value and that depends on the rating function.

In order to evaluate the recommendation performance of our traditional systems and to assess the performance contribution of its hybrid model, we used the Mahout's evaluator function. It is not possible to calculate the same by Mahout for the hybrid system because there are too many post calculations. Since the hybrid model is the combination of these two models so we can look the precision and recall value of the traditional systems:

Evaluation result for User based system:

Precision: 0.08805031446540884

Recall: 0.08720930232558138

F1 Measure: 0.08762779052785313

Evaluation result for Item based system:

Precision: 0.0

Recall: 0.0

Here Precision and Recall value is very small because we have used a very small percentage of the data set for training i.e. 0.05%. If we could use more nodes of the hadoop cluster we could have used more training data and expect a good precision and recall value.

5.2. Comparative Analysis

Look at the recommendations for user based CF recommendation systems the rating ranges from 5.0 and 3.9466 and Item based system's predicted rating is constant, 5.0. The ratings from hybrid system are ranges from 4.7121 to 4.441. Here the user based recommendation rating ranges more than the hybrid model. So, we can say that the recommended items are more closely clustered for hybrid system. On the the other hand the if we compare the item based CF result and hybrid model then we can see that the item based CF recommendation results does not have variation for rating it always rate the items the same it's a backdrop for this item based system.

Also the recommended items are different from each other. That's because the hybrid models not only combines the results of two systems it also combines the factor models which results to change of preference value.

5.3. Error Analysis

We have found some errors regarding our recommender systems. Firstly, the time taken by the system to recommend is very high. The user based approach takes around 10-15 minutes to recommend an user, The item based recommender takes around 5-10 minutes to recommend an user and the hybrid system takes 15-20 minutes to recommend an user. Reason of this problem is maybe we are using a huge data set to train the engine. The dataset

size is 1.7GB and contains 17,000,000+ rating. In future it can be resolved by utilizing hadoop with multinode cluster.

Also, the predicted rating computed by item based recommendation is 5.0. The reason is not clear but it might be mahout's framework related problem. We need to explore mahout more to resolve this matter.

The evaluator model built to evaluate the recommender systems is working properly for user based system but for item based problem it gave no value. Maybe it's also an error of the system. Apart from that the evaluator models consumes along time to evaluate the recommenders. Because of having too much post calculations we are not able to evaluate the hybrid system.

The out of memory error is most encounter error while evaluating the system. This happened because of low java heap space in operating system. This can be solving with a better high performance system.

6. Conclusion and Future Work

We have built three recommender system one follows user based collaborative filtering approach, one follows item based recommendation approach and the last one follows hybrid model. The hybrid model combines the results from user based and item based approach and also associates some factor model based on user profile to improve the recommendation mechanism. These three factor models are based on user's age, gender and country. We have recommended a user by all three recommendation system and also checked the results and compared the recommendations by all three systems.

In future we can improve the system by assigning more nodes to the system. Since, the system takes a bit time to compute the results, more nodes to hadoop can help out this situation. Also, we can add more recommendation models to the hybrid system which and improve the performance. Also we can search for some item profile based data to implement a content based model.

References

- [1] <https://Mahout.apache.org>.
- [2] <http://www.last.fm>.
- [3] Balabanović, M. and Shoham, Y., 1997. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3), pp.66-72.
- [4] Basu C, Hirsh H, Cohen W. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI 1998 Jul 26* (pp. 714-720).
- [5] Lang, K., 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995* (pp. 331-339).
- [6] Krulwich, B. and Burkey, C., 1996, March. Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the AAAI spring symposium on machine learning in information access* (Vol. 25, No. 27, p. 110).
- [7] Billsus, D. and Pazzani, M.J., 1999. A hybrid user model for news story classification. In *UM99 User Modeling* (pp. 99-108). Springer, Vienna.
- [8] Babu, M.S.P. and Kumar, B.R.S., 2011. An implementation of the user-based collaborative filtering algorithm. *International Journal of Computer Science and Information Technologies*, 2(3), pp.1283-86.
- [9] Salakhutdinov, R., Mnih, A. and Hinton, G., 2007, June. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning* (pp. 791-798). ACM
- [10] Wu, Y.H., Chen, Y.C. and Chen, A.L., 2001. Enabling personalized recommendation on the web based on user interests and behaviors. In *Research Issues in Data Engineering, 2001. Proceedings. Eleventh International Workshop on* (pp. 17-24). IEEE.
- [11] Shardanand, U. and Maes, P., 1995, May. Social information filtering: algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 210-217). ACM Press/Addison-Wesley Publishing Co..
- [12] Rucker, J. and Polanco, M.J., 1997. Siter: personalized navigation for the Web. *Communications of the ACM*, 40(3), pp.73-76.
- [13] Goldberg, D., Nichols, D., Oki, B.M. and Terry, D., 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12), pp.61-70.
- [14] Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. and Riedl, J., 1997. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3), pp.77-87.

- [15]Cotter, P. and Smyth, B., 2000. A personalized television listing service. *Communications of the ACM*, 43(8), pp.107-111.
- [16] “Spark,” available at <http://spark-project.org/>.
- [17] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, “Twister: a runtime for iterative mapreduce,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 810–818.
- [18] Uitdenbogerd, A. L. and Justin Zobel, “Manipulation of Music For Melody Matching,” *Proceedings of the sixth ACM international conference on Multimedia*, 1998, Pages 235 – 240
- [19] Mao, J. and Jain, A.K., 1992. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern recognition*, 25(2), pp.173-188
- [20] Walunj, Sachin, Sadafale, Kishor. 2013. “An online recommendation system for e-commerce based on apache Mahout framework.” *Proceedings of the 2013 annual conference on Computers and people research*, ACM (2013): 153–158.
- [21]<https://en.wikipedia.org>.
- [22]<http://www.dtic.upf.edu>.
- [23]<http://qnalist.com/questions/5329048/Mahout-vs-spark>.

Appendix A

Pre-Installations:

1. Linux OS (4 GB RAM, Core-i3 Processor)
2. OpenSSH server
3. Java
4. Hadoop
5. Mahout
6. Eclipse

The set of pre installations and configuration for this project is given below.

Openssh server Installation:

OpenSSH is a free open source set of computer programs used to provide secure and encrypted communication over a computer network by using the ssh protocol. Now we install OpenSSH Server on machine.

```
# apt-get install openssh-server
```

We can use this backup file for further configuration in future use. Now we edit this sshd_config file using any text editor and do following changes in the file :

1. Since SSH users use very weak password so it is very easy for online attackers to hack into the server. So it is better to use SSH key instead of a password. If you'll always be able to log in to your computer with an SSH key, you should disable password authentication altogether. So we need to find 'PasswordAuthentication' and change it from 'PasswordAuthentication yes' to 'PasswordAuthentication no'.

2. Allow your users who can use ssh. Add these lines (excluding the inverted commas) at the end of the file to allow the particular user (In this document we allow user 'suto') :

```
"AllowUsers suto
```

```
PermitRootLogin no
```

PubkeyAuthentication yes"

3.If you want to record all log informations like failed log in details etc. Find 'LogLevel' and Change 'LogLevel INFO' to 'LogLevel VERBOSE'.
Now save the file and restart the SSH to apply the current changes.

#systemctl restart ssh

After restarting SSH we need to generate SSH keys for user 'suto' and Since hadoop requires password each time it interacts with its nodes each time. To get rid of this job we will create RSA key without password by the following command:

\$ ssh-keygen -t rsa

Now we need to enable SSH access to the local machine by adding it to the knowing key list & test ssh.

\$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

\$ chmod 600 ~/.ssh/authorized_keys

Now we can test SSH by the following command.

\$ ssh localhost

Java Installation:

The following commands to add java repository and installing Java.

\$ add-apt-repository ppa:webupd8team/java

\$ sudo apt-get update


```
$ apt-get install oracle-java8-installer
```

Now you can check again Java is properly installed or not using ‘javac’ or ‘java -version’ command.

Hadoop Installation:

Now the environment is set for installing hadoop 2.x. In this document we will install hadoop-2.7.3. We download the tar file from following website:

Go to Hadoop website :

```
"http://www.apache.org/dyn/closer.cgi/hadoop/common/"
```

Go to Download directory from terminal and run the command for extracting the tar file.

```
$ sudo tar -xvzf hadoop-2.7.3.tar.gz
```

Then move the extracted content into Hadoop folder which is under *usr/local/hadoop* directory and make this directory available for the hadoop user account. Here our hadoop user is ‘suto’ so we make ‘suto’ owner of this directory :

```
$ sudo mv hadoop-2.7.3 /usr/local/hadoop
```

```
$ sudo chown -R suto /usr/local/hadoop
```

Update the java using the following command :

```
$ update-alternatives --config java
```

Now we need to edit ‘~/.bashrc’ file using any text editor (gedit, nano ,etc.) to configure hadoop variables ,java home variables and setting up PATH variables by adding the following lines at the end of the file. Since we are using ‘java-8-oracle’ so our JAVA_HOME must be set to that path where ‘java-8-oracle’ resides and set HADOOP_HOME to the path

where hadoop resides :

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

The directory `/usr/local/hadoop/etc/hadoop` contains configuration files. Open `hadoop-env.sh` in a text editor and set `JAVA_HOME` variable by adding the line below. This specifies the java installation that will be used by Hadoop.

```
$ cd /usr/local/hadoop/etc/hadoop
$ nano hadoop-env.sh
```

Now find `JAVA_HOME` and set the following line :

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

save and exit.

Editing the XML Based Hadoop Configuration Files :

All the Hadoop configuration files reside under `usr/local/hadoop/etc/hadoop`.

```
$ cd /usr/local/hadoop/etc/hadoop
```

Now you can check out all the xml files by 'ls' command. We are going to edit xml based configuration files:

1. Edit 'core-site.xml' by the following command and add the following texts between the configuration tags:

```
$ sudo gedit core-site.xml
```

```
<property>  
<name>fs.defaultFS</name>  
<value>hdfs://localhost:9000</value>  
</property>
```

save and exit.

2. Edit 'yarn-site.xml' by the following command and add the following texts between the configuration tags:

```
$ sudo gedit yarn-site.xml
```

```
<property>  
<name>yarn.nodemanager.aux-services</name>  
<value>mapreduce_shuffle</value>  
</property>  
<property>  
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>  
<value>org.apache.hadoop.mapred.ShuffleHandler</value>  
</property>
```

save and exit.

3. Make a copy of 'mapred.site.xml.template' with a name 'mapred.site.xml' and edit 'mapred-site.xml' by the following command and add the following texts between the

configuration tags:

```
$ sudo cp mapred.site.xml.template mapred-site.xml
```

```
$ sudo gedit mapred-site.xml
```

```
<property>  
<name>mapreduce.framework.name</name>  
<value>yarn</value>  
</property>
```

save and exit.

4. The 'hdfs-site.xml' is used to specify the namenode and datanode directories by the following command and add the following texts between the configuration tags to specify the directory paths:

```
$ sudo gedit hdfs-site.xml
```

```
<property>  
<name>dfs.replication</name>  
<value>1</value>  
</property>  
<property>  
<name>dfs.namenode.name.dir</name>  
<value>file:/usr/local/hadoop/hadoop_data/hdfs/namenode</value>  
</property>  
<property>  
<name>dfs.datanode.data.dir</name>  
<value>file:/usr/local/hadoop/hadoop_store/hdfs/datanode</value>  
</property>
```

save and exit.

Now run following commands to make directory of data node and name node.

```
$ sudo mkdir -p /usr/local/hadoop/hadoop_data/hdfs/namenode
```

```
$ sudo mkdir -p /usr/local/hadoop/hadoop_data/hdfs/datanode
```

Test Hadoop:

Format the file system by running `hdfs namenode -format` to initialize the file system.

```
$hdfs namenode -format
```

Start the single node cluster by `start-dfs.sh` and `start-yarn.sh` or we can use `start-all.sh`

```
$ start-dfs.sh
```

```
$ start-yarn.sh
```

we can check by `jps` command that all the hadoop daemons are running or not.

Eclipse Installation:

Downloading Eclipse :

We need eclipse to develop java programs so we need eclipse for java developers.

To download it go to this page: <https://eclipse.org/downloads/eclipse-packages/> and download eclipse for Linux 64bit OS and select Eclipse IDE for Java developers. Extract the tar file.

Downloading hadoop Eclipse plug-in :

Hadoop Eclipse Plug-in provides tools to ease the experience of Mapreduce on Hadoop.

Hadoop eclipse plug-in is used to associate all the hadoop accessories with eclipse. It supports to create Mapreduce and Driver classes. Also helps eclipse to browse and interacts with hdfs, submitting jobs and monitoring on their execution. To download it go to this web page:

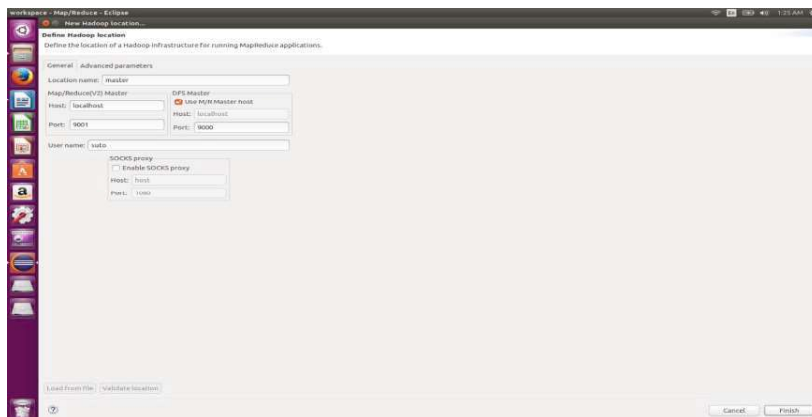
<https://github.com/Ravi-Shekhar/hadoop-eclipse-plugin/blob/master/release/hadoop-eclipse-plugin-2.6.0.jar>

After downloading the plug-in.jar file copy that file and paste this file in the folder `‘/eclipse/plugins’`.

Setting DFS Location

- Select the Mapreduce locations TAB at the bottom of the screen[fig-4]. Right click on blank space and select "New Hadoop Location".

- Give Location name e.g. "master".
- Give host name e.g. "localhost".
- Give Port number Map Reduce master= 9001 and DFS master = 9000



[Fig-4]

Mahout Installation:

Installing Maven in Eclipse:

1. *Open Eclipse -> Help -> Install new software*
2. *In text box paste this URL - <http://download.eclipse.org/technology/m2e/releases/>*
3. *Click add -> type the name [Example: m2eclipse] -> OK*
4. *Click on check box of Maven Integration for Eclipse .*
5. *Restart the Eclipse*

Maven Setup for Mahout :

Create new maven project and all the related libraries.

Edit pom.xml file

Add the Dependency inside dependencies.

```
<groupId>com.predictionmarket</groupId>
<artifactId>HybridRecommendation</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>HybridRecommendation</name>
<url>http://maven.apache.org</url>
<properties>
```

```
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
  <dependency>
    <groupId>org.apache.Mahout</groupId>
    <artifactId>Mahout-core</artifactId>
    <version>0.9</version>
    <exclusions>
      <exclusion>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-core</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-hdfs</artifactId>
    <version>2.7.3</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>2.7.3</version>
```

```
    </dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-jobclient</artifactId>
    <version>2.7.3</version>
</dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-client</artifactId>
    <version>2.7.3</version>
</dependency>
</dependencies>
</project>
```

Save and close the file

Appendix B

All the codes regarding the projects are discussed below. For the first phase we discard the data and map the strings with a unique integer as discussed above.

StringToInt.java

```
public class StringToInt {
    private Map<String, Integer> map;
    private int counter = 1;

    public StringToInt() {
        map = new HashMap<String, Integer>();
    }

    public int toInt(String s) {
        Integer i = map.get(s);
        if (i == null) {
            map.put(s, counter);
            i = counter;
            ++counter;
        }
        return i;
    }

    public int count()
    {
        return map.keySet().size();
    }

    public String toStr( int l) {
        for (String o : map.keySet()) {
            if (map.get(o).equals(l)) {
                return o;
            }
        }
    }
}
```

```

        }
        return null;
    }
}

```

MyRecommendationEngine:

```

class MyRecommendationEngine{
    static StringToInt s=new StringToInt();
    static StringToInt s1=new StringToInt();
    static StringToInt s2=new StringToInt();
    public static class Map extends Mapper<LongWritable, Text, Text,Text > {

        public void map(LongWritable key, Text value, Context context )
        throws IOException, InterruptedException {
            String line = value.toString();
            String str[]=line.split("\t");
            if(str.length==4){
                String val=str[1]+","+str[3]+","+str[2];
                Text valT=new Text();
                valT.set(val);
                context.write(new Text(str[0],valT);
            }
        }
    }
}

```

```

public static class Reduce extends Reducer<Text, Text,
Text, Text> {

    public void reduce(Text key, Iterable<Text> values,
Context context) throws IOException, InterruptedException {
        ArrayList<Integer> obj = new ArrayList<Integer>();
        Iterator<Text> values1=values.iterator();

```

```

ArrayList<Text> cache=new ArrayList<Text>();
while( values1.hasNext() ) {
    Text val=values1.next();
    String str[]=val.toString().split(",");
    try{
        obj.add((Integer.parseInt(str[1])));
        cache.add(new Text(val));
    }catch(Exception e){
        System.out.println(e);
    }
}
int dif=Collections.max(obj)-Collections.min(obj);
long size=cache.size();
for (int i = 0; i < size ; ++i){
    Text val=cache.get(i);
    double rate;
    String str[]=val.toString().split(",");
    try{
        if(dif==0)
            rate=2.5;
        else{
            rate=1+(((float)Integer.parseInt(str[1]) -
(float)Collections.min(obj))/dif)*4;
        }
        Integer k=s.toInt(key.toString());
        Integer k1=s1.toInt(str[0]);
        s2.toInt(str[2]);
        context.write(new Text(k+","+k1+","+rate),new Text(""));
    }
    catch(Exception e){
        System.out.println(e);
    }
}
}

```

```

    }
}

public static void RecomanderFile(String user) throws TasteException, IOException
{
    Int userid=s1.toInt(user);
    PrintWriter pw = new PrintWriter(new File("user_song"));
    pw.write("\n"+i+". Recommendation for User "+s.toStr((int)i)+" :\n");
    List <RecommendedItem>recommendations = App.recom(userid);
    for (RecommendedItem recommendation : recommendations) {
        pw.write("Recommended Artist
        :"+s2.toStr((int)recommendation.getItemID())+"    Predicted
        Rating:"+recommendation.getValue()+"\n");
    }
    System.out.println("STOP RECOMMENDATION");
pw.close();
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    @SuppressWarnings("deprecation")
    Job job = new Job(conf, "categories");
    job.setJarByClass(MyRecommendationEngine.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
}

```

```

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
Path out=new Path(args[1]);
out.getFileSystem(conf).delete(out);
job.waitForCompletion(true);
RecomanderFile();
}
}

```

Here the RecommenderFile(String) recommend the user and take argument user's mboxsha1 id. It recommend based on the App.recom(int);

Here I user three type of recommendation system in side App classes.

User Based Collaborative filtering

```

public class App
{
    public static List <RecommendedItem> recom(int UserID ) throws IOException,
TasteException{
        DataModel model = new FileDataModel(new File("data/part-r-00000"));
        UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
        UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1,
similarity, model);
        UserBasedRecommender recommender = new
GenericUserBasedRecommender(model, neighborhood, similarity);
        List <RecommendedItem>recommendations =
recommender.recommend(UserID,20);
        return recommendations;
    }
}

```

Item Based Collaborative Filtering:

```

public class App
{

```

```

        public static List <RecommendedItem> recom(int UserID ) throws IOException,
TasteException
        {
            DataModel model = new FileDataModel(new File("data/part-r-00000"));
            ItemSimilarity similarity = new PearsonCorrelationSimilarity(model);
            ItemBasedRecommender recommender = new
GenericItemBasedRecommender(model, similarity);
            List <RecommendedItem>recommendations =
recommender.recommend(UserID,20);
            return recommendations;
        }
    }
}

```

In our final model i.e. the hybrid mode is consists of many parts one of the parts is the App.java which combines the user based and item based suggestions and along with that there are three factor models which is build based some mathematical calculations discussed above in detail.

Hybrid recommendation engine:

```

public class App
{
    private static GenderBasedDataModel sModel = null;
    private static AgeBasedDataModel aModel = null;
    private static CountryBasedModel cModel = null;
    private static DataModel model = null;
    private static String GenderBasedModelPath = "data/store2/usersha1-profile.tsv";
    private static String AgeBasedModelPath = "data/store2/usersha1-profile.tsv";
    private static String CountryBasedModelPath = "data/store2/usersha1-profile.tsv";
    private static String DataModelPath = "data/store1/part-r-00000";
    public static void Init(StringToInt s) throws TasteException, IOException{
        try {
            sModel = new GenderBasedDataModel(GenderBasedModelPath,s);
        } catch (IOException e) {

```

```

        System.out.println("Gender Based Data not available. Skipping this
data.");
    }
    try {
        aModel = new AgeBasedDataModel(AgeBasedModelPath,s);
    } catch (IOException e) {
        System.out.println("Gender Based Data not available. Skipping this
data.");
    }
    try {
        cModel = new CountryBasedModel(CountryBasedModelPath,s);
    } catch (IOException e) {
        System.out.println("Gender Based Data not available. Skipping this
data.");
    }
    try {
        model = new FileDataModel(new File(DataModelPath));
    } catch (IOException e) {
        System.out.println("Data is not available. Exiting...");
        System.exit(1);
    }
}
}

```

```

    public static ArrayList<StringBuilder> recom(int UserID) throws IOException,
TasteException{
        UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
        ItemSimilarity similarity1 = new PearsonCorrelationSimilarity(model);
        UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1,
similarity, model);
        UserBasedRecommender recommender = new
GenericUserBasedRecommender(model, neighborhood, similarity);

```

```

        ItemBasedRecommender recommender1 = new
GenericItemBasedRecommender(model, similarity1);
        List <RecommendedItem>recommendations =
recommender.recommend(userID,100);
        List <RecommendedItem>recommendations1 =
recommender1.recommend(userID,100);
        Map<RecommendedItem, Double> itemWeights = new
HashMap<RecommendedItem, Double>();
        double max=0;
        for(RecommendedItem r : recommendations){
            if(max< (double)r.getValue())
                max=(double)r.getValue();
        }
        for(RecommendedItem r : recommendations){
            itemWeights.put(r, ((double)r.getValue()*5.0/max)*0.7);
        }
        for(RecommendedItem r : recommendations1){
            itemWeights.put(r, (double)r.getValue()*0.7);
        }
        factorModels(userID, itemWeights);
        Set<Entry<RecommendedItem, Double>> entrySet = itemWeights.entrySet();
        List<Entry<RecommendedItem, Double>> entryList = new
ArrayList<Entry<RecommendedItem, Double>>(entrySet);
        Collections.sort(entryList, new Comparator<Entry<RecommendedItem,
Double>>(){
            @Override
            public int compare(Entry<RecommendedItem, Double> o1,
Entry<RecommendedItem, Double> o2)
            {
                return o2.getValue().compareTo(o1.getValue());
            }
        });

```



```

ArrayList<StringBuilder> res=new ArrayList<StringBuilder>();
for(int i=0; i<20; i++){
    StringBuilder sb=new StringBuilder();
    sb.append(entryList.get(i).getKey().getItemID());
    sb.append(',');
    sb.append(entryList.get(i).getValue());
    res.add(sb);
    System.out.println(sb+"\n");
}
return res;
}

```

```

private static void factorModels(int userID, Map<RecommendedItem, Double>
weights) throws TasteException {
    for(RecommendedItem r : weights.keySet()){
        if(sModel != null) {
            PreferenceArray preferences =
model.getPreferencesForItem(r.getItemID());
            factorGenderModel(userID, weights, r, preferences);
        }
        PreferenceArray preferencesForItem =
model.getPreferencesForItem(r.getItemID());
        if(aModel != null){
            factorAgeModel(userID,weights, r, preferencesForItem);
        }
        if(cModel != null)
        {
            factorCountryModel(userID,weights, r, preferencesForItem);
        }
    }
}
}

```

```

private static void factorCountryModel(int userID, Map<RecommendedItem,
Double> weights, RecommendedItem r, PreferenceArray preferences) {
    int tcount=0;
    int ucount=0;
    double pcr=0;
    double pref=0;
    double ucc=cModel.getUserCountryCount(userID);//user country count
    double acc=cModel.getAllCountryCount();//all country count
    double acr=ucc/acc;//all country ratio
    double advr=0;
    for(Preference p : preferences)
    {
        tcount++;

        if(cModel.getCountryByUserID(p.getUserID()).equals(cModel.getCountryByUserID(userID)
        ))
            ucount++;
    }
    pcr=((double)ucount)/tcount;
    advr=pcr/acr;
    pref=(advr*advr/(1.0+advr*advr));
    weights.put(r, weights.get(r) +pref*.5);
}

```

```

private static void factorGenderModel(int userID, Map<RecommendedItem,
Double> weights, RecommendedItem r, PreferenceArray preferences) {
    int mgender = 0;
    int fgender = 0;
    double pref = 0;
    double advr=0,mr=0,fr=0;
    int tmc=sModel.getMaleGendercount();

```

```

int tfc=sModel.getFemaleGendercount();
for(Preference p : preferences)
{
    if(p.getValue() > 0)
    {

if(sModel.getGenderTypeByUserID(p.getUserID()).equalsIgnoreCase("m"))
        {
            mgender++;
        }
        else
if(sModel.getGenderTypeByUserID(p.getUserID()).equalsIgnoreCase("f"))
        {
            fgender++;
        }

    }
}
mr=(double)mgender/tmc;
fr=(double)fgender/tfc;
    if(sModel.getGenderTypeByUserID(userID).equalsIgnoreCase("f"))
    {
        if(tfc==0||mr==0)
            weights.put(r, weights.get(r) +pref*.5);
        else{
            advr=((double)fr)/(mr);
            pref=(advr*advr/(1.0+advr*advr));
            weights.put(r, weights.get(r) +pref*.5);
        }
    }
else if(sModel.getGenderTypeByUserID(userID).equalsIgnoreCase("m"))
    {
        if(tmc==0||fr==0)

```

```

        weights.put(r, weights.get(r) +pref*.5);
    else{
        advr=((double)mr)/(fr);
        pref=(advr*advr/(1.0+advr*advr));
        weights.put(r, weights.get(r) +pref*.5);
    }
}
}
}

```

Now here are the codes for the three factor models part of the hybrid system:

Gender Based Model:

```

public class GenderBasedDataModel {

    private String path;
    private Map<Long, UserGender> modelData;

    public GenderBasedDataModel(String path, StringToInt s) throws IOException
    {
        File file = new File(path);
        if(!file.exists() || file.isDirectory())
        {
            throw new FileNotFoundException(file.toString());
        }
        else
        {
            this.path = path;
        }
        modelData = new HashMap<Long, UserGender>();
        buildModel(s);
    }

    private void buildModel(StringToInt s) throws IOException
    {
        try(CSVReader reader = new CSVReader(new FileReader(path), '\t'))

```

```

{
String[] line;
    while ((line = reader.readNext()) != null) {
        if (line.length >= 4) {
            if (line[1].equals("m")||line[1].equals("f")) {
                long id = s.toInt(line[0]);
                modelData.put(id, new UserGender(line[1]));
            }
        }
    }
}

```

```

public String getGenderTypeByUserID(long id)
{
    if(modelData.containsKey(id))
    {
        return modelData.get(id).getGender();
    }
    return "undef";
}

```

```

public int getMaleGendercount()
{
    Set<Long> all = modelData.keySet();
    int count=0;
    for(long i:all)
    {
        if(getGenderTypeByUserID(i) != null){
            if(getGenderTypeByUserID(i).equalsIgnoreCase("m")){
                count++;
            }
        }
    }
}

```

```
    }  
    return count;  
}  
  
public int getFemaleGendercount()  
{  
    Set<Long> all = modelData.keySet();  
    int count=0;  
    for(long i:all)  
    {  
        if(getGenderTypeByUserID(i) != null){  
            if(getGenderTypeByUserID(i).equalsIgnoreCase("f")){  
                count++;  
            }  
        }  
    }  
    return count;  
}  
  
private class UserGender  
{  
    private String gender;  
    public UserGender( String gender)  
    {  
        this.gender = gender;  
    }  
  
    public String getGender() {  
        return gender;  
    }  
    public void setGender(String gender) {  
        this.gender = gender;  
    }  
}
```

```

    }

}

private static void factorAgeModel(int userID, Map<RecommendedItem, Double>
weights, RecommendedItem r, PreferenceArray preferences) {
    int uage = aModel.getAgeTypeByUserID(userID);
    int page = 0;
    int difage=0;
    int count=0;
    double sum=0,sum1=0;
    double var=0;
    double cov=0;
    double mean=0;
    double pref=0;
    if(uage!=-1){
        System.out.println(userID+"age"+pref+"|"+(weights.get(r) +pref*.5));
        weights.put(r, weights.get(r) + .25);
        return ;
    }
    for(Preference p : preferences)
    {

        page=aModel.getAgeTypeByUserID(p.getUserID());
        if(page!=-1)
            continue;
        difage=uage-page;
        sum=sum+(difage*difage);
        sum1=sum1+page;
        count++;
    }
}

```

```

        // System.out.println("CU"+userID+"|"+uage+"
VU"+p.getUserID()+"|"+page+"|"+difage+"|"+sum1/count+"|"+Math.sqrt(sum/count));

    }

    var=Math.sqrt(sum/count);
    mean=sum1/count;
    cov=var/mean;
    pref=1-cov;
    System.out.println(userID+"age"+pref+"|"+(weights.get(r) +pref*.5));
        weights.put(r, weights.get(r) + pref*.5);
    }
}

```

Country Based Model:

```

public class CountryBasedModel {
    private String path;
    private Map<Long, UserCountry> modelData;
    public CountryBasedModel(String path, StringToInt s) throws IOException
    {
        File file = new File(path);
        if(!file.exists() || file.isDirectory())
        {
            throw new FileNotFoundException(file.toString());
        }
        else
        {
            this.path = path;
        }
        modelData = new HashMap<Long, UserCountry>();
        buildModel(s);
    }
}

```



```

private void buildModel(StringToInt s) throws IOException
{
    try(CSVReader reader = new CSVReader(new FileReader(path), '\t'))
    {
        String[] line;
        while ((line = reader.readNext()) != null) {
            if (line.length >= 4 && !line[3].equals("")) {
                long id = s.toInt(line[0]);
                modelData.put(id, new UserCountry(line[3]));
            }
        }
    }
}

public int getUserCountryCount(long UserID)
{
    Set<Long> all = modelData.keySet();
    int count=0;
    for(long i:all)
    {
        if(getCountryByUserID(i) != null){
            if(getCountryByUserID(i).equalsIgnoreCase(getCountryByUserID(UserID))){
                count++;
            }
        }
    }
    return count;
}

public int getAllCountryCount()
{
    Set<Long> all = modelData.keySet();
    int count=0;
    for(long i:all)

```

```
{
    if(getCountryByUserID(i) != null){
        count++;
    }
}
return count;
}

public String getCountryByUserID(long id)
{
    if(modelData.containsKey(id))
    {
        return modelData.get(id).getCountry();
    }
    return "undef";
}

private class UserCountry
{
    private String con;
    public UserCountry( String con)
    {
        this.con = con;
    }

    public String getCountry() {
        return con;
    }

    public void setCountry(String con) {
        this.con = con;
    }
}
```

```
}
```

Age Based Factor Model:

```
public class AgeBasedDataModel {
    private String path;
    private Map<Long, UserAge> modelData;
    public AgeBasedDataModel(String path, StringToInt s) throws IOException
    {
        File file = new File(path);
        if(!file.exists() || file.isDirectory())
        {
            throw new FileNotFoundException(file.toString());
        }
        else
        {
            this.path = path;
        }
        modelData = new HashMap<Long, UserAge>();
        System.out.println("here 1");
        buildModel(s);
    }
}
```

```
private void buildModel(StringToInt s) throws IOException
{
    try(CSVReader reader = new CSVReader(new FileReader(path), '\t'))
    {
        String[] line;
        while ((line = reader.readNext()) != null) {
            if (line.length >= 4 && !line[2].equals("")) {
                try{
                    int age=Integer.parseInt(line[2]);
                    long id = s.toInt(line[0]);
                    if(age>=0&&age<=100)
                        modelData.put(id, new UserAge(age));
                }
            }
        }
    }
}
```

```
        } catch(NumberFormatException e)
        {
            System.out.println(e);
        }
    }
}
}
```

```
public int getAgeTypeByUserID(long id)
{
    if(modelData.containsKey(id))
    {
        return modelData.get(id).getAge();
    }
    return -1;
}
```

```
private class UserAge
{
    private int Age;
    public UserAge( int age2)
    {
        this.Age = age2;
    }

    public int getAge() {
        return Age;
    }
    public void setAge(int Age) {
        this.Age = Age;
    }
}
```

```
}

```

We use IRStatisticsEvaluator to evaluate our Recommendation system and also we compute Precision, Recall and F1, the Evaluation class is given below :

Recommendation Evaluator:

```
public class EvaluateRecommender {

    public static void main(String[] args) throws IOException, TasteException {
        DataModel model = new FileDataModel(new File("data/part-r-00000"));
        RecommenderIRStatsEvaluator evaluator = new
GenericRecommenderIRStatsEvaluator();
        RecommenderBuilder builder = new MyRecommenderBuilder();
        IRStatistics result=evaluator.evaluate(builder, null, model, null, 5,
GenericRecommenderIRStatsEvaluator.CHOOSE_THRESHOLD, 0.0005);
        PrintWriter pw = new PrintWriter(new File("EvaluationRecommender"));
        pw.write("START EVALUATION"+"\\n");
        pw.write("Precision :"+result.getPrecision()+"\\n");
        pw.write("Recall :"+result.getRecall()+"\\n");
        pw.write("F1 Measure"+result.getF1Measure()+"\\n");
        System.out.println("STOP EVALUATION");
        pw.close();

    }
}

```

Here

evaluate(RecommenderBuilder recommenderBuilder, DataModelBuilder dataModelBuilder, DataModel dataModel, IDRescorer rescorer, int at, double relevance Threshold, double evaluation Percentage) is to evaluate the system. The "at" value, as in "precision at 5". For example, this would mean precision evaluated by removing the top 5 preferences for a user

and then finding the percentage of those 5 items included in the top 5 recommendations for that user.
