

AUTOMATIC AUDIO CLIPS RANKING SYSTEM USING COLLABORATIVE FILTERING ALGORITHM: IMPLEMENTATION

¹RACHANA & ²ASTT. PROF. MS SHILPA KHURANA

^{1&2} CBS Group of Institutions, Maharshi Dayanand University, Haryana

ABSTRACT

Collaborative filtering (CF) is a strategy utilized by recommender systems [1]. A key issue of collective sifting is the way to join and weight the inclinations of client neighbors. At times, clients can quickly rate the prescribed things. The proposed work should achieve proactive information driven approach for cutting edge suggestion and prescient framework. The information that the associations are attempting to comprehend is immersed with heaps of commotion information called infused information however Shilling Attack. In the proposed work information thought about is gathered from group of content records Collaborative separating frameworks have many structures, however numerous basic frameworks can be decreased to two stages: 1. Look for clients who share a similar rating designs with the dynamic client (the client whom the expectation is for). 2. Use the evaluations from those similarly invested clients found in step 1 to compute an expectation for the dynamic user. The proposed framework is securing recommender frameworks against shilling assaults.

1. INTRODUCTION:

Collaborative filtering (CF) is a technique used by recommender systems.[1] .The growth of the Internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering. Collaborative filtering is one of the techniques used for dealing with this problem.

The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with tastes similar to themselves. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis. Collaborative filtering methods recommend items

based on users' past preferences, new users will need to rate sufficient number of items to enable the system to capture their preferences accurately and thus provides reliable recommendations.

Collaborative filtering algorithms often require:

- (1) Users' active participation,
- (2) An easy way to represent users' interests, and
- (3) Algorithms that is able to match people with similar interests.

Typically, the workflow of a collaborative filtering system is:

1. A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.
2. The system matches this user's ratings against other users' and finds the people with most "similar" tastes.
3. With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user.

2. OBJECTIVE

To deal with too many ratings can be an issue, for most collaborative filtering systems, having to deal with too few ratings is a far more serious problem. This problem occurs when the amount of items become very large reducing the number of items users have rated to a tiny percentage. In such a situation it is likely that two people have few rated items in common making the correlation coefficient less reliable. This is called the sparsity problem. In proposed work, the solutions have been proposed to overcome this problem:

- Implicit ratings: Some systems, such as a later extension of GroupLens [Miller et al. 1997], try to increase the number of ratings by inferring them from the user's behavior. However, a user still has to explore an item before the system can infer a rating.
- Dimensionality reduction: By reducing the dimensionality of the information space, the ratings of two users can be used for predictions even if they did not rate the same items. Pazzani and Billsus [Pazzani&Billsus 1998] use an non correlation-based approach for making predictions based on a neural network.
- Content description: Using the content of an item instead of the item itself could increase the amount of information people have in common. This is a hybrid approach to combining content-based filtering and collaborative filtering and is described in the next section in more detail.

3. ABOUT HADOOP AND COLLABORATIVE FILTERING:

As the quantities of clients and things develop, customary CF calculations will endure genuine versatility issues. For instance, with countless clients and a great many things, a CF calculation with the unpredictability of is now too huge. So the proposed framework is utilizing Hadoop Platform. In a proposal framework where everybody can give the appraisals, individuals may give bunches of positive evaluations for their own particular things and negative evaluations for their rivals. Assailants could infuse countless profiles, called shilling profiles or bots. In the proposed work of shared separating frameworks precautionary measures are acquainted with debilitate such sort of controls. Shilling assaults possibly hurt the recommender framework and in this way they are accepting increasingly consideration alongside the expanding significance of RS. The proposed work is utilizing Hadoop.

Hadoop: The Hadoop disseminated record framework (HDFS) is a device to actualize dispersed registering and parallel handling for grouping and questioning information accumulations. Preparing extensive datasets with Hadoop has indicated great outcomes. Hadoop is a structure that backings data intensive appropriated applications; it enables applications to keep running on a great many PCs and can diminish handling time altogether. Hadoop has been utilized to take care of an assortment of issues with conveyed processing, for instance, machine interpretation on extensive information [4, 10, 11] and assessing the distance across of vast diagrams [7]. HDFS was roused by two systems from Google, MapReduce [3] and Google File System (GFS) [5]. GFS is an adaptable appropriated record framework for information concentrated applications and gives a blame tolerant approach to store information on ware equipment. GFS can convey high total execution to an expansive number of customers. The MapReduce show, made by Google, gives a straightforward and capable interface that empowers programmed parallelization and dissemination of substantial calculations on ware PCs. The GFS gives MapReduce circulated capacity and high throughput of information. Hadoop is the open source usage of GFS and MapReduce.

4. LITERATURE REVIEW:

Shilling attacks potentially harm the recommender system and therefore they are receiving more and more attention along with the increasing importance of RS. Lam and Riedl explored several open questions regarding the effectiveness of shilling attacks (RandomBot and AverageBot)^[1]. They found that user-based CF algorithms were less tolerant to shilling attacks than item-based CF algorithms. Watching for sharp changes in the value of traditional algorithm performance metrics such as mean absolute error (MAE) may be useful for detecting some attacks. However, many effective attacks will not be visible through simple aggregate metrics like MAE. In addition, new or obscure items, particularly in the user-based CF algorithm, are especially susceptible to attack. Mobasher et al. studied six different attack models and measured their effectiveness in both user-based and item-

based CF^[13]. The prediction shift and hit ratio were used to measure how attacks can affect the RS. Their results show that the different types of attacks can effectively and practically harm the standard CF algorithms. The same authors in ^[14] presented a formal framework for specifying attack models and attack profiles and introduced a classification approach for attack detection. Their work mainly focused on showing the effectiveness of an attack, rather than the detection of attacks. Particularly, a type of segmented attack was studied. Unlike RandomBot and AverageBot that have no special target users, this segmented attack pushes an item to a targeted group of users with known or easily predicted preferences. Profiles are inserted that maximize the similarity between the pushed item and items preferred by the group. The segmented attack is both effective and practical against standard item-based CF algorithms. The detection of shilling attack has also been studied. Su et al. worked on finding the group shilling, which is a type of coalition shilling attack^[4]. They constructed a bipartite graph for the users and items, and used similarity spreading algorithm to find user clusters. They then labeled one cluster to be an abnormal group of shilling users, if the size (number of members) and average similarity of this cluster with the other clusters were smaller than pre-defined thresholds. With the assumption that shillers work together and they are highly correlated, Mehta used probabilistic latent semantics analysis (PLSA) and principal component analysis (PCA) to eliminate the clusters of shilling users^[2]. PLSA, a soft clustering method, computes a probabilistic distribution over communities (clusters of users) based on latent factors. It has been employed to remove much of the influence of biased attack profiles for model-based systems in ^[3]. PCA, a linear dimensionality reduction model, was used to select dimensions which are very different, or as in this work, very similar to other dimensions. The intuition of identifying the community (cluster of users) to be removed is that the cluster containing shilling profiles will be tighter, leading to lower average distances from the centroid of the cluster. Therefore, average Mahalanobis distance of a community was used to examine how closely knit a community is. Zhang et al. detected the attacks by treating the ratings for an item as a time series according to their given time^[5]. The sample average and sample entropy within a disjoint window of k consecutive ratings are calculated to capture the change in an item's likability and the distributional change in an item's ratings.

5. IMPLEMENTATION ISSUES:

XYZ.com is an online music site where clients tune in to different tracks, the information gets gathered like demonstrated as follows. A guide decrease program is composed to get Number of one of kind audience members. The information document contains million of records. The information is coming in log documents and looks like as demonstrated as follows:

UserId|TrackId|Shared|Radio|Skip

111115|222|0|1|0

111113|225|1|0|0

111117|223|0|1|1

111115|225|1|0|0

The proposed work will take care of the main issue that is discovering novel audience members per track from the enormous information, information of million records.

Most importantly it is have to comprehend the information, here the main section is UserId and the second one is Track Id. So it is required to compose a mapper class which would produce trackId and userIds and middle of the road key esteem sets. To make it easy to recall the information arrangement, it is required make a constants class.

It is required to expel invalid record or infused records.

A Reducer class would be made to total the outcomes. Here essentially whole reducer couldn't be utilized as the records are getting are not one of a kind and we need to number just one of a kind clients.

Shilling Attack could be arranged by disposing of copy userIds and making a Driver class.ud computing.

6. PROPOSED WORK:

The proposed work should achieve proactive data driven approach for advanced recommendation and predictive system.

Despite the fact that there is an enormous spike accessible data, the rate of the information that could be understand and utilized further is on decline.

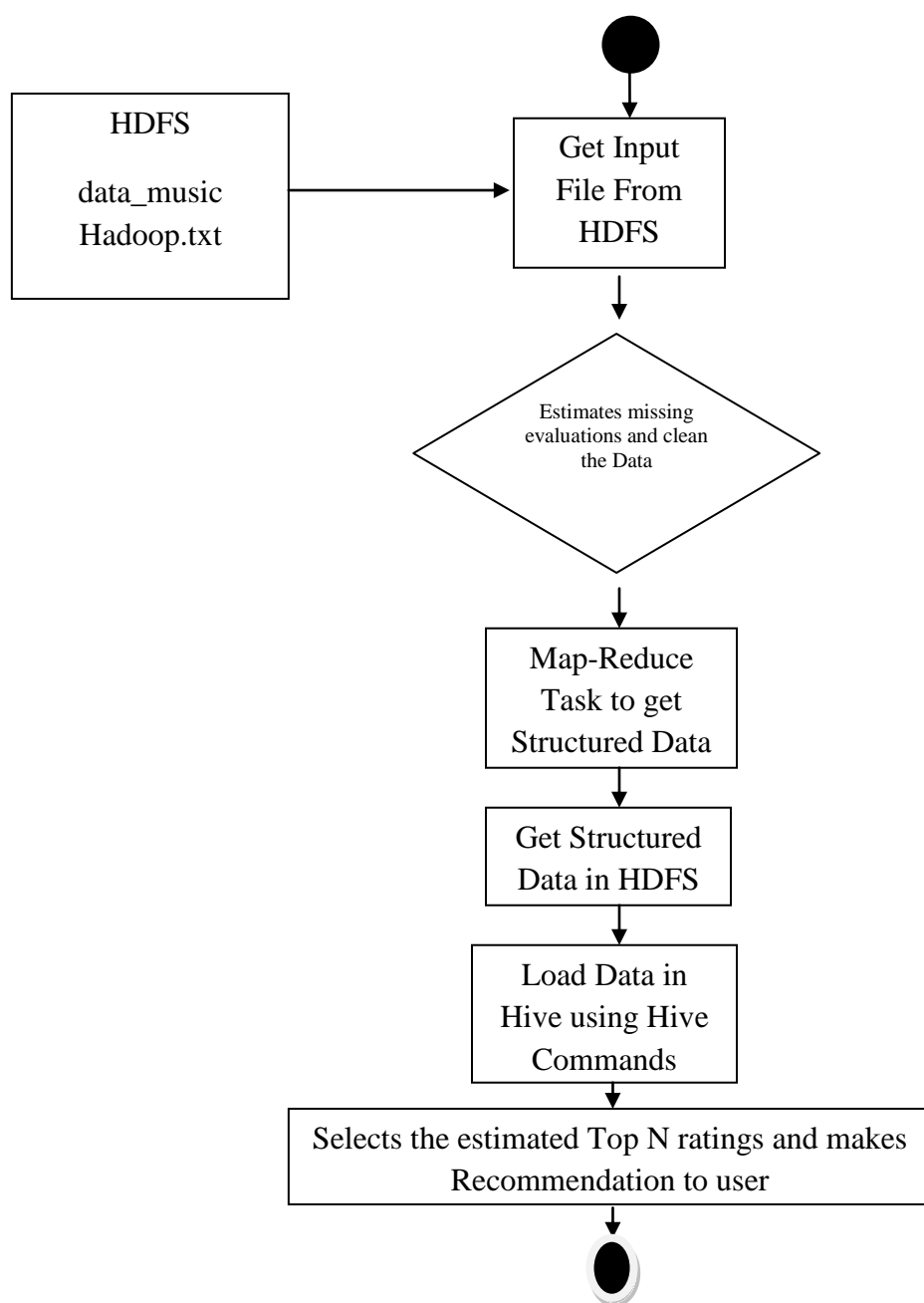
The data that the organizations are attempting to understand is saturated with lots of noise data called injected data though Shilling Attack.

As the quantities of users and items increase, conventional CF algorithms will suffer genuine scalability issues. For instance, with a huge number of users and a huge number of items, a CF algorithm with the complexity of is as of now too extensive. So the proposed framework is utilizing Hadoop Platform.

In a recommendation framework where everybody can give the ratings, individuals may give loads of positive evaluations for their own items and negative appraisals for their rivals. Aggressors could infuse an extensive number of fake profiles, called shilling profiles or bots. In the proposed work of collaborative filtering systems precautions are applied to manage such sort of manipulations. Shilling attack conceivably hurt the recommender framework and therefore they are receiving more and more attention along with the increasing importance of RS.

Flow of the Recommendation Engine

Figure 1.2: Flow Chart



<u>Track Id</u>	<u>User ID</u>
T1	U1
T2	U2
T3	U1
T2	U7
T4	U3
T1	U3
T2	U8
T4	U6
T2	U9
T4	U8
T1	U2

Table: Track Ids with their Listeners Id

According to the above data

Track Id T1 is listened by U1, U2, U3

Track Id T2 is listened by U2, U7, U8, U9

Track Id T3 is listened by U1

Track Id T4 is listened by U6, U8

So T2 can be recommended as it has maximum 4 listeners.

T1 can also be recommended as it has 3 listeners.

T3 has minimum listeners.

7. IMPLEMENTATION:

The HDFS namespace is stored by the NameNode. The NameNode utilizes an exchange log called the EditLog to relentlessly record each change that jumps out at document framework metadata. For instance, making another document in HDFS causes the NameNode to embed a record into the EditLog demonstrating this. So also, changing the replication component of a document makes another record be embedded into the EditLog. The NameNode utilizes a record in its neighborhood have OS document framework to store the EditLog. The whole record framework namespace, including the mapping of squares to documents and document framework properties, is put away in a record called the FsImage. The FsImage is put away as a document in the NameNode's nearby record framework as well.

The NameNode keeps a picture of the whole record framework namespace and document Blockmap in memory. This key metadata thing is intended to be conservative, to such an extent that a NameNode with 4 GB of RAM is bounty to bolster a colossal number of records and catalogs. At the point when the NameNode begins up, it peruses the FsImage and EditLog from circle, applies every one of the exchanges from the EditLog to the in-memory portrayal of the FsImage, and flushes out this new form into another FsImage on plate. It can then truncate the old EditLog on the grounds that its exchanges have been connected to the industrious FsImage. This procedure is known as a checkpoint. In the present usage, a checkpoint just happens when the NameNode begins up. Work is in advance to bolster occasional check pointing sooner rather than later.

The Data Node stores HDFS information in records in its nearby document framework. The DataNode has no information about HDFS records. It stores each square of HDFS information in a different record in its neighborhood document framework. The DataNode does not make all documents in a similar registry. Rather, it utilizes a heuristic to decide the ideal number of records per catalog and makes subdirectories suitably. It is not ideal to make every neighborhood document in a similar registry in light of the fact that the nearby record framework won't not have the capacity to productively bolster an immense number of documents in a solitary index. At the point when a DataNode begins up, it look over its nearby record framework, produces a rundown of all HDFS information obstructs that compare to each of these neighborhood documents and sends this answer to the NameNode: this is the Blockreport.

EXPERIMENTAL IMPLEMENTATIONS

XYZ.com is an online music site where users listen different tracks, the information gets gathered like shown as follows. A Map-Reduce program is composed to get Number of unique. The information document contains million of records.

The information is coming in log documents and looks like as demonstrated as follows:

UserId|TrackId|Shared|Radio|Skip

111115|222|0|1|0

111113|225|1|0|0

111117|223|0|1|1

111115|225|1|0|0

In the initial segment of the work, we will tackle the main issue that is discovering extraordinary audience members per track.

As a matter of first importance it is have to comprehend the information, here the primary segment is UserId and the second one is Track Id. So it is required to compose a mapper class which would emanate trackId and userIds and transitional key esteem sets. To make it easy to recollect the information grouping, it is required make a constants class.

It is required to expel invalid record or infused records.

A Reducer class would be made to total the outcomes. Here essentially whole reducer couldn't be utilized as the records are getting are not remarkable and we need to number just novel clients.

Shilling Attack could be arranged by disposing of copy userIds and making a Driver class

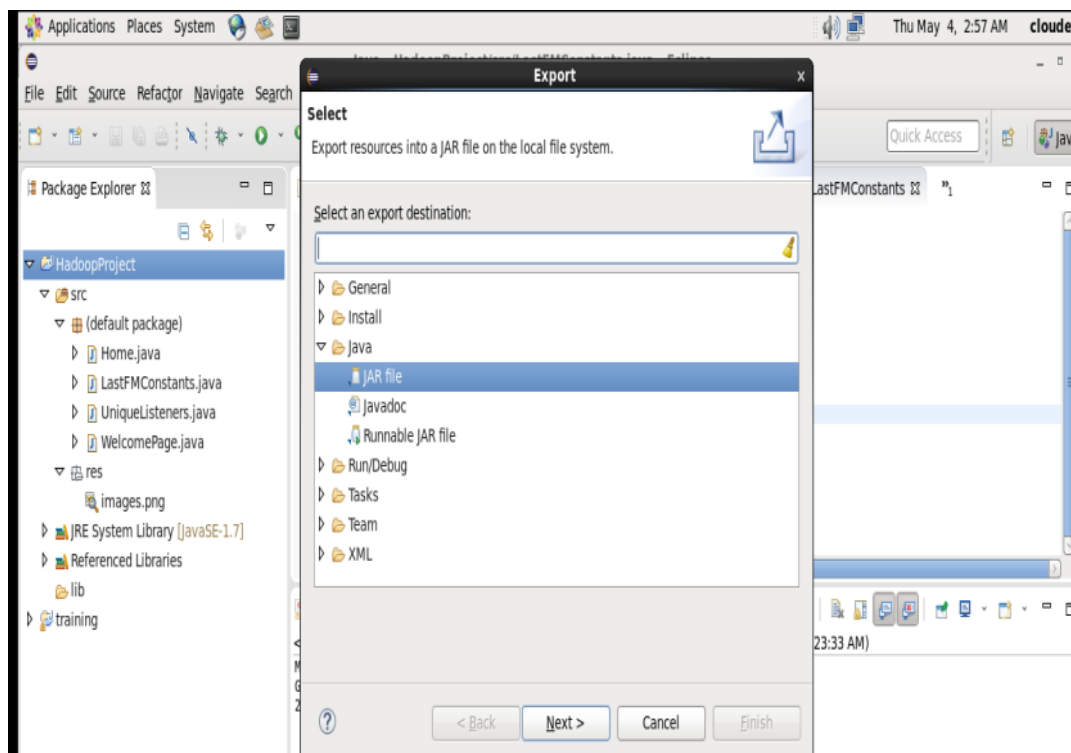
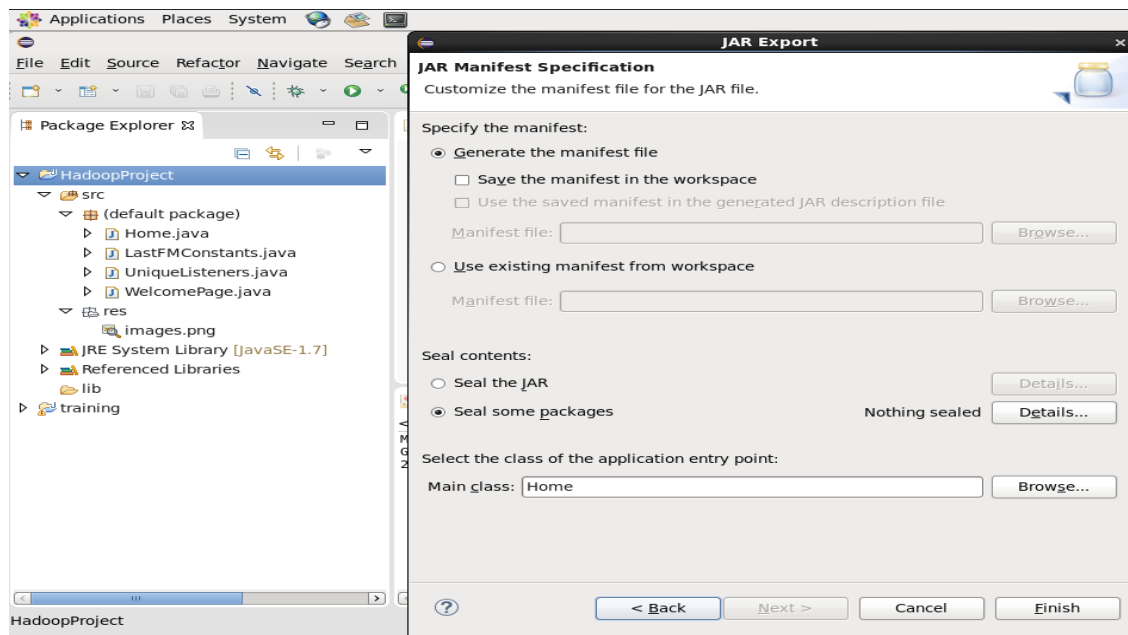


Figure: To create jar file

A java project is created in Eclipse in Cloudera. This project contains four java files Home.java, Welcome Page. java , LastMFConstants.java and UniqueListeners.java. The Home.java is to get Splash screen. And Splash.jar is created and saved in Cloudera home, to run this java file. This java program is using Threads to wait the screen for five seconds and then open WelcomePage.Java.



The Hadoop Command used to run Splash.jar to get Home.java

Figure: *hadoop jar Splash.jar Home*

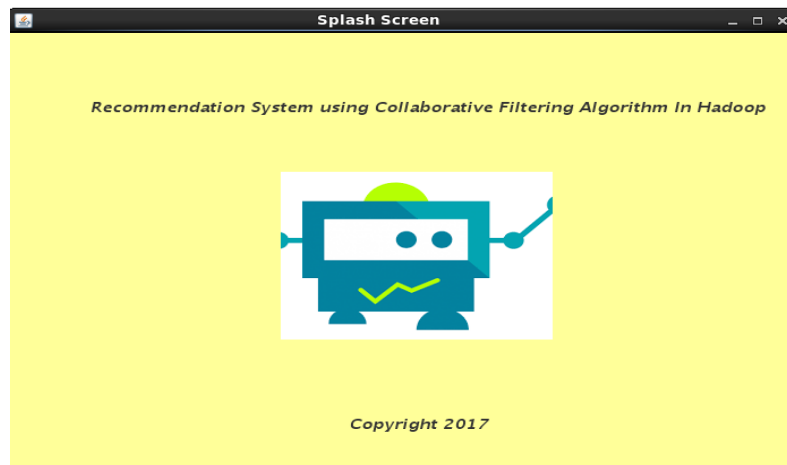


Figure: Splash Screen

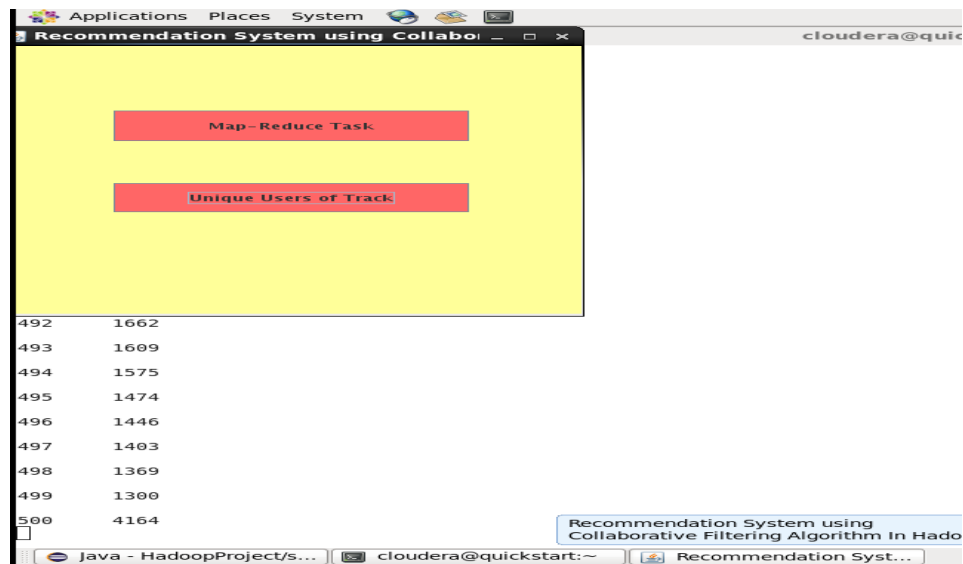


Figure: Total users for each track

WelcomePage contains two buttons. The first map-reduce button performs the Map-Reduce task using track.jar and a hadoop command. The track.jar is already created from the java HadoopProject in eclipse.

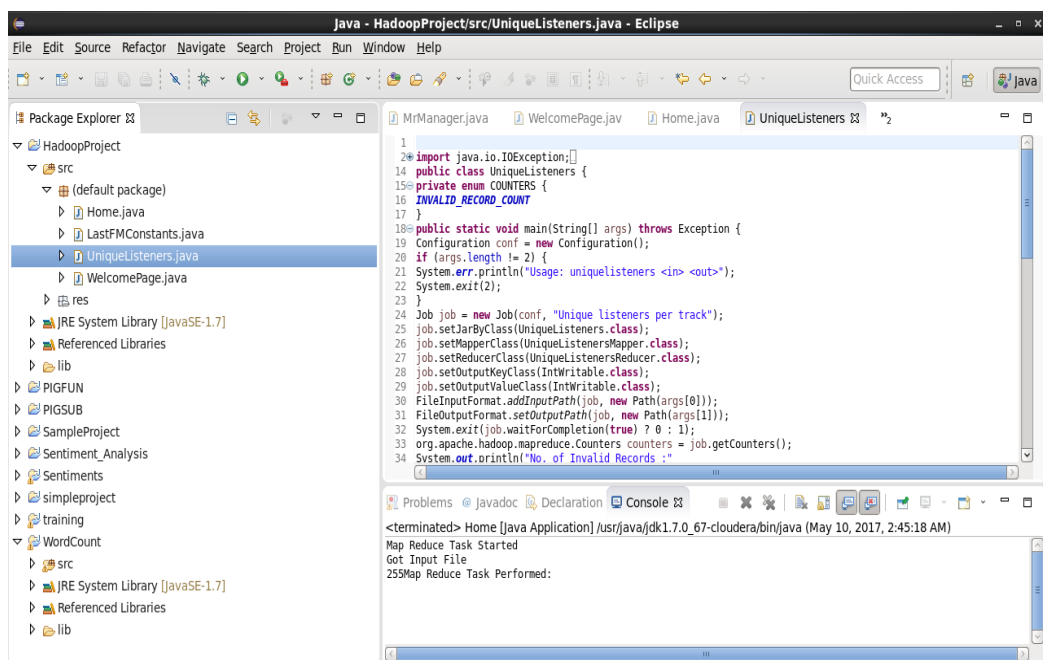
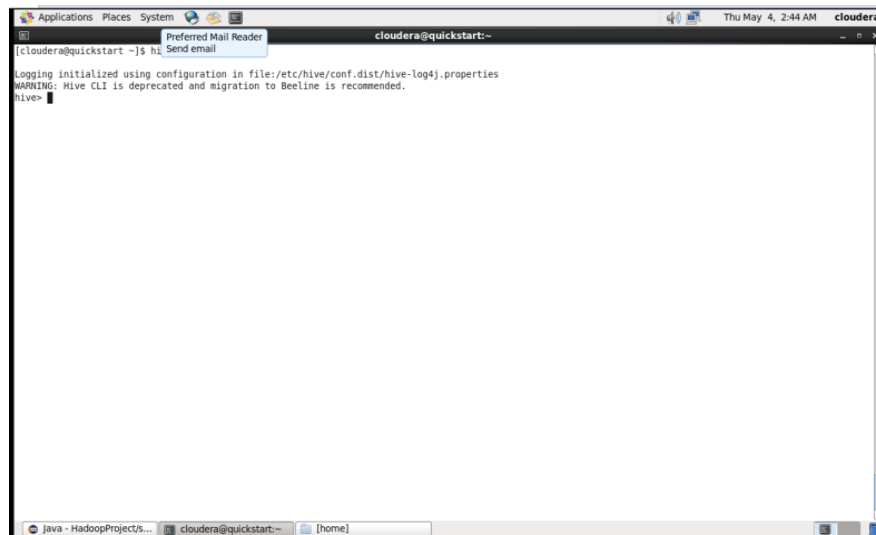


Figure:hadoop jar track.jar input/ output

The input is the folder contains data_musicHadoop.txt After Map-Reduce task in hadoop ,the output file is loaded in the hive. Open the Terminal and give Hive Command.

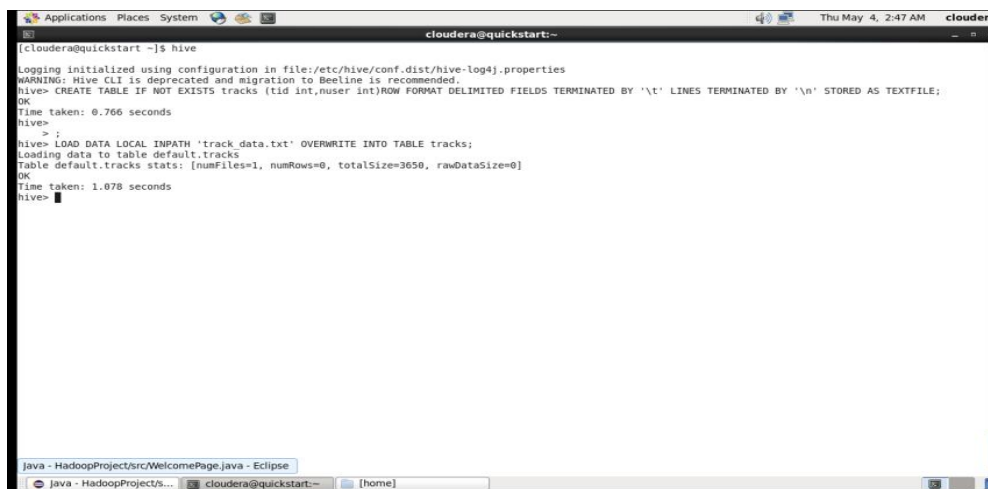


```

Applications Places System cloudera@quickstart:~
[cloudera@quickstart ~]$ hi Send email
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>

```

Figure: Hive Command

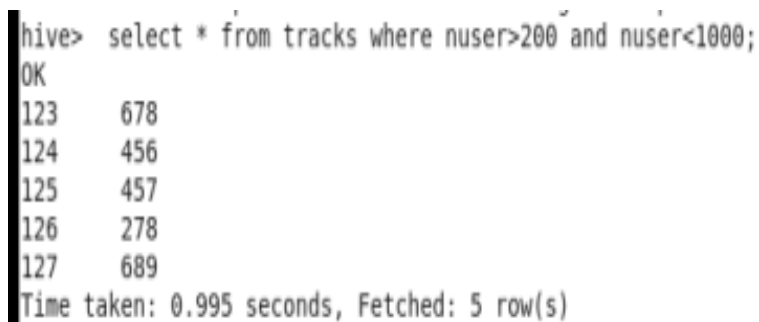


```

Applications Places System cloudera@quickstart:~
[cloudera@quickstart ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE TABLE IF NOT EXISTS tracks (tid int,nuser int)ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' STORED AS TEXTFILE;
OK
Time taken: 0.766 seconds
hive>
>
hive> LOAD DATA LOCAL INPATH 'track data.txt' OVERWRITE INTO TABLE tracks;
Loading data to table default.tracks
Table default.tracks stats: [numFiles=1, numRows=0, totalSize=3650, rawDataSize=0]
OK
Time taken: 1.078 seconds
hive>

```

Figure: Creating Table and Loading Data in Hive



```

hive> select * from tracks where nuser>200 and nuser<1000;
OK
123      678
124      456
125      457
126      278
127      689
Time taken: 0.995 seconds, Fetched: 5 row(s)

```

Figure:Hive Command.

To recommend top 10 tracks, we can use hive command.

```
hive> select * from tracks where nuser<4828 and nuser>4800;
OK
286      4818
287      4817
291      4821
292      4810
293      4801
295      4811
296      4805
298      4806
299      4804
300      4813
302      4801
303      4805
304      4815
305      4817
307      4814
309      4820
315      4820
316      4804
Time taken: 0.087 seconds, Fetched: 18 row(s)
```

Figure: Top 15 records.

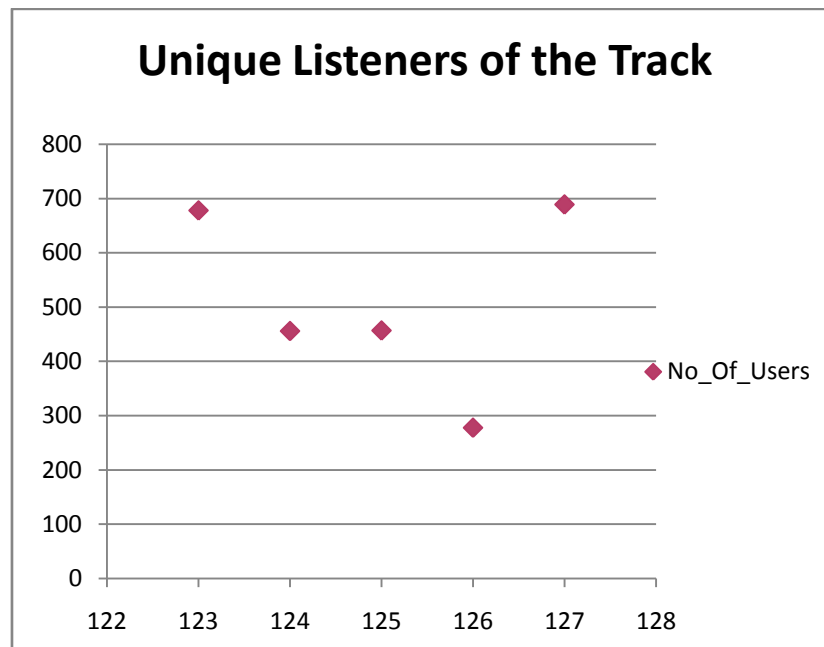


Figure: Maximum users of the Tracks

Above graph shows that the tracks having Id 291,315 and 316 could be recommended. They have maximum listeners.

```
hive> select * from tracks where nuser>200 and nuser<1000;
OK
123      678
124      456
125      457
126      278
127      689
Time taken: 0.995 seconds, Fetched: 5 row(s)
```

Figure: Tracks having minimum users

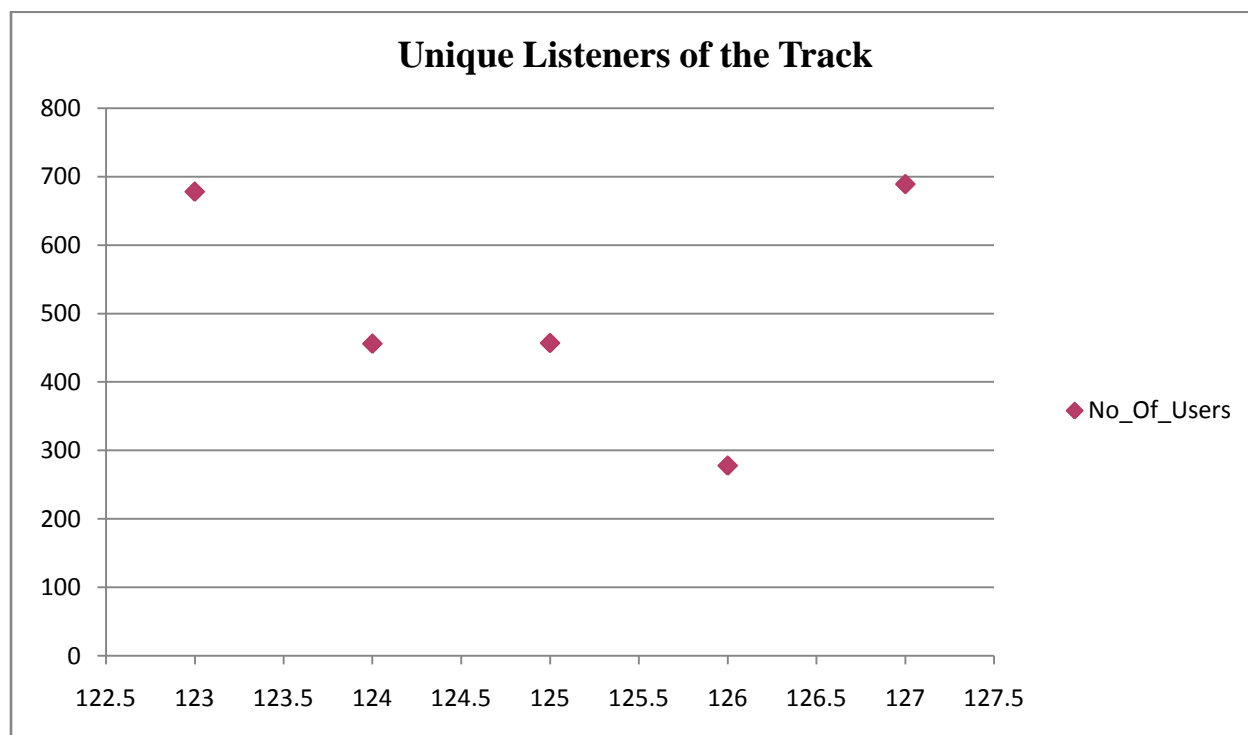


Figure: Tracks having minimum users.

7. MATERIAL AND METHODS UTILIZED:

- Algorithms used:

1. Memory-based Collaborative Filtering Algorithms
2. Model-based Collaborative Filtering Algorithms

- Minimum requirement:

1. Input text files – log file
2. Virtual Machine and Cloud era
3. Eclipse to create the mapper, reducer and driver classes. To process the input files in Hadoop Environment

- Most Common HDFS operations used:

Creating Directory

Removing Directory

Copying files to/from HDFS

List Content of Directory

Contents of File

Analyze space allocation

Check Permissions/write ability

8. FUTURE SCOPE:

Recommendation systems will work in internet business to offer a more instinctive, immersive and balanced involvement for each progression of a customer's journey. The development of the Internet has made it considerably more hard to effectively extract valuable data from all the accessible online data. The staggering measure of information requires systems for productive data separating. Recommendation systems have the impact of controlling clients personally to intriguing articles in an extensive space of conceivable choices. In the proposed work collaborative filtering is utilized for RS. The collaborative filtering includes substantial informational collections. The collaborative filtering can be utilized further to make making automatic predictions about the interests of a user by gathering preferences or taste data from numerous clients by means of collaboration.

9. CONCLUSION:

The proposed framework is ensuring recommender frameworks against shilling assaults. The calculation can be utilized for observing client evaluations and expelling shilling assailant profiles from the way toward registering suggestions, in this way keeping up the high caliber of the proposal.

10. REFERENCE:

- [1] Arthur D, Vassilvitskii S. k-means++: The advantages of careful seeding. In Proc. the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, Jan. 2007, pp.1027-1035.
- [2]. DuBois T, Golbeck J, Kleint J, Srinivasan A. Improving recommendation accuracy by clustering social networks with trust. In Proc. ACM Workshop on Recommender Systems & the Social Web, Oct. 2009.
- [3] Jamali M, Ester M. TrustWalker: A random walk model for combining trust-based and item-based recommendation. In Proc. the 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Jun. 2009, pp.397-406.
- [4] Lam S K, Riedl J. Shilling recommender systems for fun and profit. In Proc. the 13th International Conference on World Wide Web (WWW), May 2004, pp.393-402.
- [5] Ma H, King I, Lyu M R. Learning to recommend with social trust ensemble. In Proc. the 32nd Int. ACM SIGIR Conf. Research and Develop. in Inform. Retrieval, Jul. 2009, pp.203-210.

- [6] Ma H, Lyu M R, King I. Learning to recommend with trust and distrust relationships. In Proc. the 3rd ACM Conference on Recommender Systems, Oct. 2009, pp.189-196.
- [7] Marsh S P. Formalising trust as a computational concept [Ph.D. Thesis]. University of Stirling, Apr. 1994.
- [8] Massa P, Avesani P. Trust-aware recommender systems. In Proc. the 1st ACM Conference on Recommender Systems, Oct. 2007, pp.17-24.
- [9]. Mehta B. Unsupervised shilling detection for collaborative filtering. In Proc. the 22nd National Conference on Artificial Intelligence, Jul. 2007, pp.1402-1407.
- [10].Mobasher B, Burke R, Bhaumik R, Williams C. Effective attack models for shilling item-based collaborative filtering systems. In Proc. WebKDD Workshop, Aug. 2005.
- [11] Mobasher B, Burke R, Bhaumik R, Williams C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Trans. Internet Technol., Oct. 2007, 7(4): Article No.23
- [12].Mobasher B, Burke R D, Sandvig J J. Model-based collaborative filtering as a defense against profile injection attacks. In Proc. the 21st National Conference on Artificial Intelligence, Jul. 2006, pp.1388-1393
- [13] O'Connor M, Herlocker J. Clustering items for collaborative filtering. In Proc. SIGIR 2001 Workshop on Recommender Systems, Sept. 2001.
- [14] O'Donovan J, Smyth B. Trust in recommender systems. In Proc. the 10th International Conference on Intelligent User Interfaces, Jan. 2005, pp.167-174..
- [15] Pitsilis G, Zhang X L, Wang W. Clustering recommenders in collaborative filtering using explicit trust information. In Proc. the 5th IFIP WG 11.11 International Conference on Trust Management (IFIPTM), Jun. 2011, pp.82-97.
- [16] Rodgers J, Nicewander A. Thirteen ways to look at the correlation coefficient. The American Statistician, 1988, 42: 59-66.
- [17]Sarwar B M, Karypis G, Konstan J, Riedl J. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In Proc. the 5th Int. Conf. Computer and Information Technology, Dec. 2002
- [18] Su X F, Zeng H J, Chen Z. Finding group shilling in recommendation system. In Special Interest Tracks and Posters of the 14th WWW, May 2005, pp.960-961
- [19]. Truong K Q, Ishikawa F, Honiden S. Improving accuracy of recommender system by item clustering. Transactions on Information and Systems, 2007, E90-D(9): 1363-1373
- [20] Zhang S, Chakrabarti A, Ford J, Makedon F. Attack detection in time series for recommender systems. In Proc. the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 2006, pp.809-814.