Implementation and Comparison of Recommender Systems using various models

Ankan Mazumdar
Dept of Computer Science
NSEC, WBUT, Kolkata

June 20, 2017

Abstract:

Recommender systems are tools for interacting with large and complex information spaces. They provide a personalized view of such spaces, prioritizing items likely to be of interest to the user. Recommender systems research has incorporated a wide variety of artificial intelligence techniques including machine learning, data mining, user modeling, case-based reasoning, and constraint satisfaction, among others. Personalized recommendations are an important part of many on-line e-commerce applications such as Amazon.com, Netflix, and Pandora. This wealth of practical application experience has provided inspiration to researchers to extend the reach of recommender systems into new and challenging areas.The goal of a Recommender System is to generate meaningful recommendations to a collection of users for items or products that might interest them. We here aim to do a comparative study among the various prevalent recommendation system models anddetermine the pros and cons along with the reasons thereof of each of these models.

Introduction:

A recommender system can be defined as “Any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options” (Burke 2002). Recommender systems form a specific type of information filtering (IF) technique that attempts to present information items (movies, music, books, news images, web pages, etc.) that are likely of interest to the user. Typically, a recommender system compares the user's profile to some reference characteristics, and seeks to predict the 'rating' that a user would give to an item they had not yet considered.

The myriad approaches to Recommender Systems can be broadly categorized as -

- **Collaborative Filtering (CF):** In CF systems a user is recommended items based on the past ratings of all users collectively.
- **Content-based recommending:** These approaches recommend items that are similar in content to items the user has liked in the past, or matched to attributes of the user.
- **Hybrid approaches:** These methods combine both collaborative and content based approaches.

Dataset Used:

We have used the widely popular MovieLens 100k dataset for trying out our systems. The reason for using this dataset is that the data is made available by a reputed and trustworthy source (University of Minnesota) and contains the information about the movies and the demographic data for users which we require in our content-based analysis. The dataset contains 100,000 ratings from 943 users on 1682 movies. Each user has rated at least 20 movies and information is also given on the movies including the title, cast, release, date and its genre.

Data Analysis:

We analyzed the given to dataset to get some insight. Here are some of the results:

- The pie chart in Figure 1 displays the overall distribution of ratings. We observed that the percentage of people voting for 3, 4, 5 is unexpectedly higher that people voting for 1 and 2. This concludes that people have tendency to not rate a movie rather than rating it low if they do not like it.

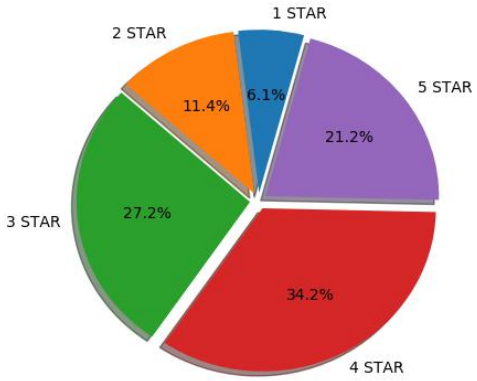


Figure 1: Distribution of User Ratings

- Scatter plot in Figure 2 displays the mean rating v/s variance scatter plot. We observe from the plot that very high rating (around rating 5) or very low rating (around rating 1) tend to have low variance compared to rating in middle range. So, we can assume end rating cases as consistent among different users.

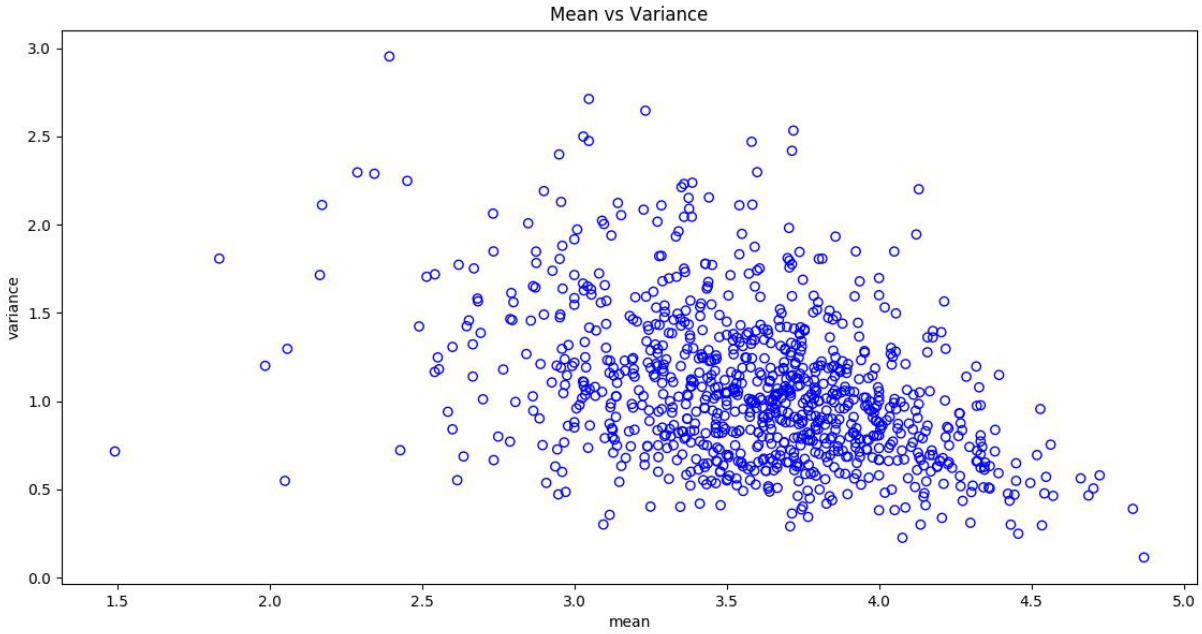


Figure 2: Scatter plot of Mean vs Variance of User Ratings

- Figure 3 is a full plot representing number of ratings given by every user. We observe that minimum number of ratings given by any user is 20 and maximum number of ratings by any user is 737. We also observe that on an average number of ratings given by user is around 100. Hence we have a considerable density of ratings in the whole range of data set.

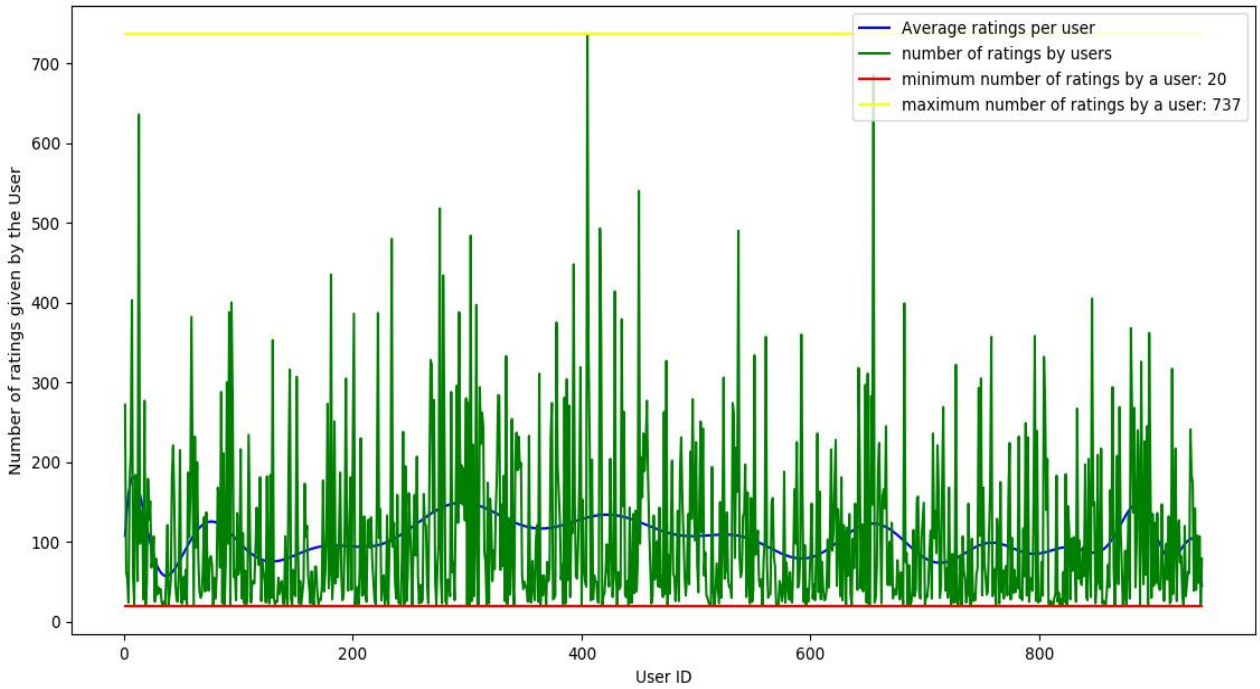


Figure 3: Statistical data on number of ratings per user

Hardware and Software Used:

- 2 Machines :

  - Processor : Intel® Core™ i5-2400 CPU @ 3.10GHz x 4

  - RAM : 12 Gb

  - Storage : 500 Gb

- Hadoop:

  - Cluster: 3 Nodes (1 NameNode and 2 Data Nodes)

  - Version: Hadoop 2.8.0

  - Components: Pig, Hive, Sqoop

- MySQL Server:

  - Version: 7-May-2017 Release

- MongoDb:

  - Version: 3.4.5

- Apache Tomcat:

  - Version: 7

Models:

- **Content Based Model -** A pure content-based recommender system makes recommendations for a user based solely on the profile built up by analyzing the content of items which that user has rated in the past. This method suggest items to users based on content information of items for example in our case we can suggest movies based on genre similar to genre that a particular user generally prefers. We used Item Profile and User Profile to suggest Items to users.

  Item Profile: I = vector of size 18 (no. of Genres) with 1 or 0 marking the presence of a Genre.

  User Profile:

  1) Find average rating given by each user.
  2) Subtract this average from all ratings of a user to get Normalized Ratings.
  3) Suppose a user rated 'n' movies with 'A' genre, then profile weight of genre 'A' for that user will be

  $$U_A = (r_1 + r_2 + r_3 + \ldots + r_n) / n$$

  where $r_1, r_2, r_3, \ldots r_n$ are Normalized Ratings for that user.

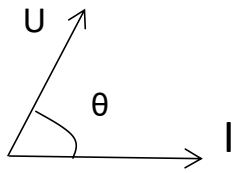
Similarly find U_B , U_C … to get the User Profile for that user as U = [U_A U_B U_C ….. 18]

Predicting:

For Recommending items to a user we will find a similarity parameter (Cosine Similarity) for all the items which user has not rated. Then Recommend top 'h' items, with high similarity parameter, to that user.

Cosine Similarity:

$$\text{Similarity Parameter} = \cos\theta = \frac{U \cdot I}{|U| * |I|}$$



which actually is the Dot Product of vectors U and I.

- **Collaborative Filtering Model -** The Collaborative Filtering Process relies upon trying to predict the opinion the user will have on the different items and be able to recommend the "best" items to each user based on the user's previous likings and the opinions of other like minded users. It is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. A person who wants to see a movie for example, might ask for recommendations from friends. The recommendations of some friends who have similar interests are trusted more than recommendations from others. Most collaborative filtering systems apply the so called neighborhood-based technique. In the neighborhood-based approach a number of users is selected based on their similarity to the active user. A prediction for the active user is made by calculating a weighted average of the ratings of the selected users. Here we have used item-item model.

  Consider there are 'U' users and 'I' items.

  1) Let Vi for item i be a vector of size U, with values as rating given by users (0 if not given).
  2) Calculate average of this vector Vi for each item, not count '0's ratings, i.e. if 'n' users gave rating to item 'i', add all the ratings and divide the sum by 'n'.
  3) Subtract this average from all the non zero values in the vector Vi, to get the normalized vector.
  4) Similarly calculate this Normalized Vector for each item.
  5) Now we need to calculate the similarity of an item with all the other items.
     Let V_a , V_b be the Normalized Vectors of items 'a' and 'b', then similarity between them can be calculated as:

$$\text{Sim}(a, b) = \frac{V_a \cdot V_b}{|V_a| * |V_b|}$$

6) After you have calculated similarity of an item 'i' with all the other items, select top 'N' items similar to 'i', i.e. with high value of Sim(a,b). {The 'N' nearest neighbours to 'i'}

7) After you have found 'N' nearest neighbours for all the items, the predicted rating of User 'x' for item 'i' is given by:

$$R(x, i) = \frac{\sum_{j \in N(i,x)} Sim(i,j) * R(x,j)}{\sum_{j \in N(i,x)} Sim(i,j)}$$

where ,

R(x,j) = Rating given by User 'x' to item 'j'.

N(i,x) = Set of items rated by 'x' similar to 'i', i.e. the set of N neighbours which we found earlier.

Sim(i,j) = Similarity between item 'i' and 'j'.

- **Hybrid Model -** We tried to build a simple model to verify our claim that a hybrid model could perform better than individual methods. We took a linear combination of these techniques - content-based model (Baseline Predictor) and collaborative filtering model. We estimate a baseline prediction for the unknown rating of User 'u' for item 'i'.

$$b_{ui} = b_i + b_u + \mu$$

where

b_i is the item bias

b_u is the user bias

μ is the average rating of dataset

b_i and b_u calculate the deviation of item i and user u from the average item and user ratings in the complete dataset.

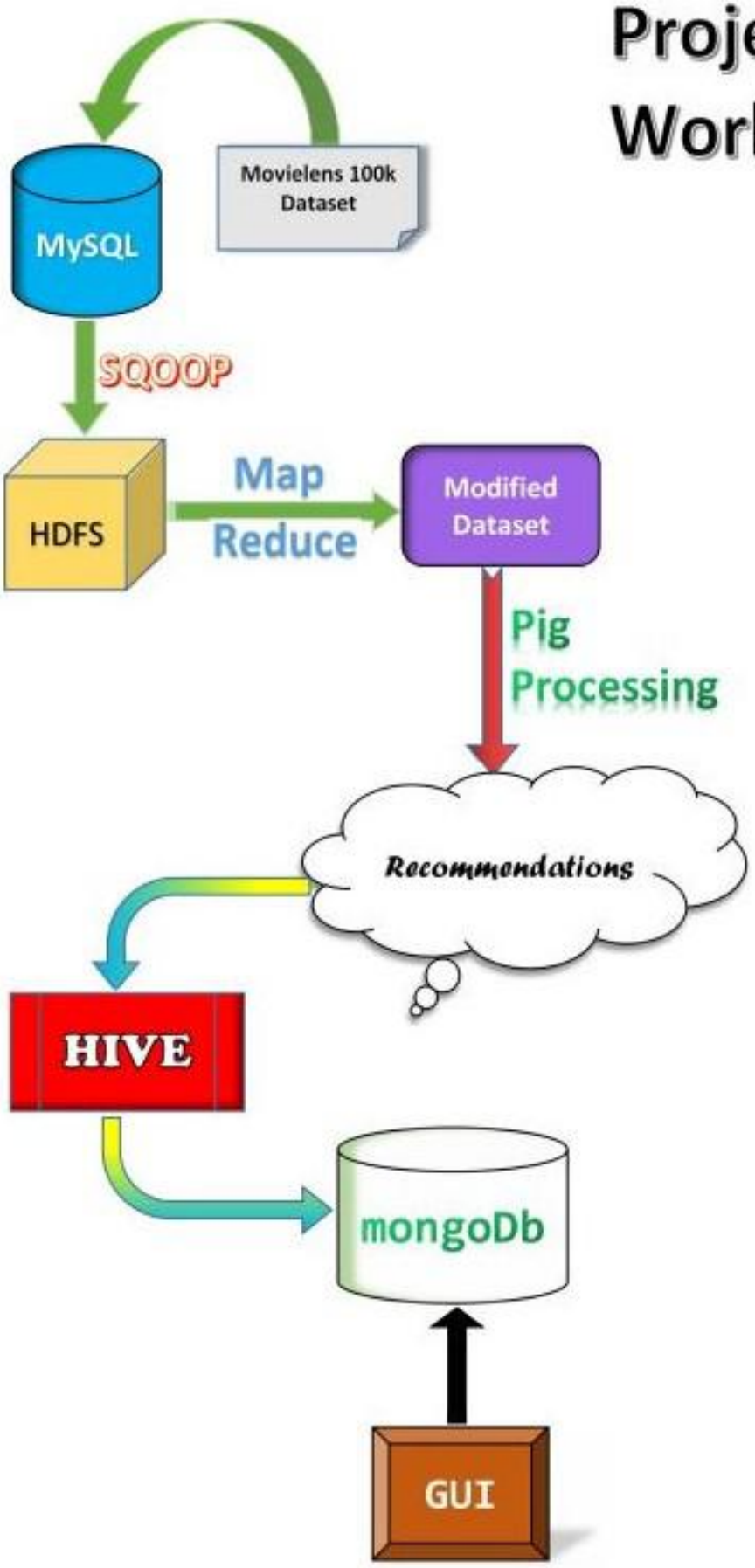
b_i = rating deviation of item 'i' = (avg. rating for item 'i') - μ

b_u = rating deviation of user 'u' = (avg. rating for user 'u') - μ

Then Predicted rating for User 'x' of item 'i' will be:

$$R(x, i) = b_{xi} + \frac{\sum_{j \in N(i,x)} Sim(i,j) * (R(x,j) - b_{xj})}{\sum_{j \in N(i,x)} Sim(i,j)}$$

Project Work-flow



Work-flow Description

Work-flow of this project can be divided into 7 main steps. This work-flow was designed assuming the environment that we will have to face in an industry. But first things first, follow the following directory structure to stay away from confusions.

➢ HDFS Directory Structure :

A) Directories you need to create:-

/RcomSys

- /OriginalData (Contains u.data file which is -> User Item Rating)
- /Items (Contains u.item -> Item details)
- /User (Contains u.user -> User details)
- /Genre (Contains u.genre -> Genre details)

B) Directories that will be automatically created:-

/RcomSys

- /UserItemRating (Output of MapReduce Job -> Modifeid Dataset)
- /ContentOut (Output of ContentBased.pig)
- /CollabOut (Output of Collaborative.pig)
- /HybridOut (Output of Hybrid.pig)

➢ Steps :

1) Creating the Environment

Component Used: MySQL

Assuming that a customer has stored his data on MySQl Server, we placed our data on MySQL Server too and then did the further processing.

2) Storing Data on HDFS

Component Used: SQOOP

As we did our whole processing on Hadoop Framework, we need to place our data from MySQL Server to HDFS (Hadoop Distributed File System). Component used to perform this operation was SQOOP. A single line of code is enough to copy data from MySQL Server Database to HDFS.

Command:

sqoop import --connect jdbc:mysql://localhost/<mysqlDbName> --table <mysqlTableName> --m 1 --fields-terminated-by '\t' --target-dir /RcomSys/OriginalData;

3) Modifying Data

Component Used: Map Reduce

The original downloaded data contains ratings of Movies which were rated by the user to the Movies he/she has viewed. But this data does not contain any rating for the Movies which were not rated by user. Our task in this step is to assign 0 (Zero) rating to the Movies which were not rated by user. This step was performed according to the requirements of the Algorithms we are using.

The code for Driver, Mapper and Reducer Classes of Map Reduce Job is attached with this report.

Command:

hadoop jar DSModification.jar PDriver /RcomSys/OriginalData/u.data /RcomSys/UserItemRating

4) Processing Data

Component Used: Pig

This is the most important step in the whole project as it contains the implementation of the algorithm which we are using. As stated above, we have used three models for generating recommendations. We wrote one Pig Script for each Model which takes Modified Data as input from HDFS. For each of 948 users we recommended 5 Movies which were generated following the Algorithms.

3 Pig Scripts corresponding to each Model has been attached with this report. Input data for these Pig scripts will be the output of Map Reduce Job in Step-3. That is, we need to set the Output directory of Step-3 as input directory for Step-4.

Command:

pig ContentBased.pig

pig Collaborative.pig

pig Hybrid.pig

5) Further Processing/Querying Output

Component Used: Hive

For querying the Output (Recommendations- 5 per User) in order to get more details about the Movies which we are Recommending to a user, we placed our output of Step-4 in Hive Tables. For eg; if you want to which genre is most like by a particular user, then you can easily write Hive queries on these tables to get the desired information.

We actually placed all our data, i.e. User Ratings, User Details, Movie Details, etc ; every thing on Hive Tables so that we can easily Query out out desired output from this processed data.

A Hive Script (hivescript1.sql) has been attached with this report which will create all the required tables.

Command:

hive -f hivescript1.sql

6) Placing the output on mongoDb

Component Used: mongoDb and Hive

As most of the companies these days are using mongoDb over MySQL to store their data because of its NoSQL properties, we also exported all the data in our Hive table into mongoDb database creating collections corresponding to each Hive table.

We need to start mongoDb Server and run the attached Hive Script (hivescript2) to export data from Hive to mongoDb.

Command: First place the attached Hive Jars into Hive's lib folder and then run the following command.

hive -f hivescript2.sql

7) Creating GUI

Component Used: JSP and mongoDb

Depicting the industrial environment and to show our Output, we created a GUI using JSP over mongoDb. Some Screen-shots from the GUI are give at the last.

Full web application is attached with this report.

Result:

Verifying the Output

We removed 100 ratings from the original data set and stored them separately. Now we predicted these removed ratings using Collaborative Filtering and Hybrid model. For comparing we calculated the Root Mean Squared Error in the Predicted Ratings with respect to the original ratings. Results are as follows:

rmse Collaborative Filtering : 1.0038

rmse Hybrid Model : 1.0013

Conclusion:

We studied all the three models individually and observed that each model has its own strengths.

- Hybrid Model turns out to be the best recommender out of three method we used, with an average **RMSE** of **1.0013**
- Collaborative Filtering Model with an **average RMSE** of **1.0038** and could provide limited personalization.
- Content Based Model with an average RMSE of 0.9999999999 stands at the last of all the three models.

References:

- Video Lectures by Dr. Anand Rajaraman , Stanford University, on Mining of Massive Data-sets.

  https://www.youtube.com/watch?v=1JRrCEgiyHM

  https://www.youtube.com/watch?v=h9gpufJFF-0

  https://www.youtube.com/watch?v=6BTLobS7AU8

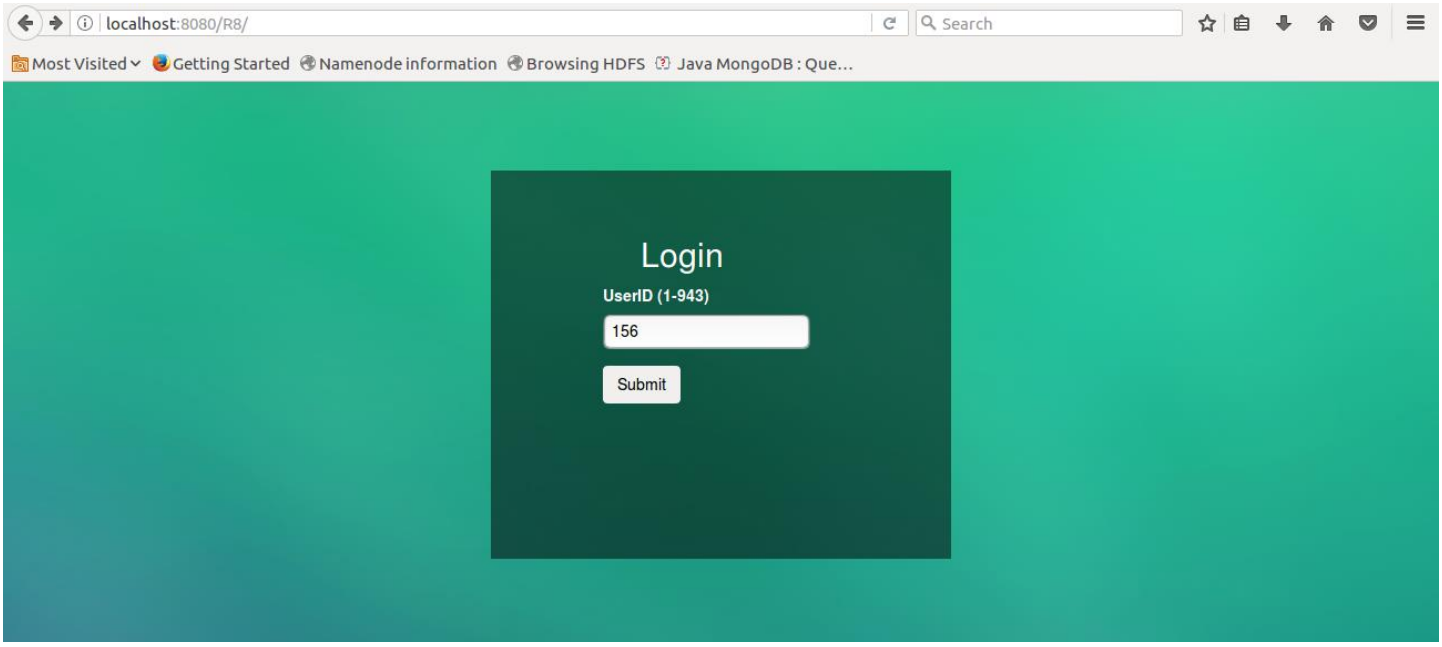
  https://www.youtube.com/watch?v=VZKMyTaLI00

- Machine Learning Course at Coursera by Dr. Andrew Ng , former Professor at Stanford University, currently Chairman of Coursera.

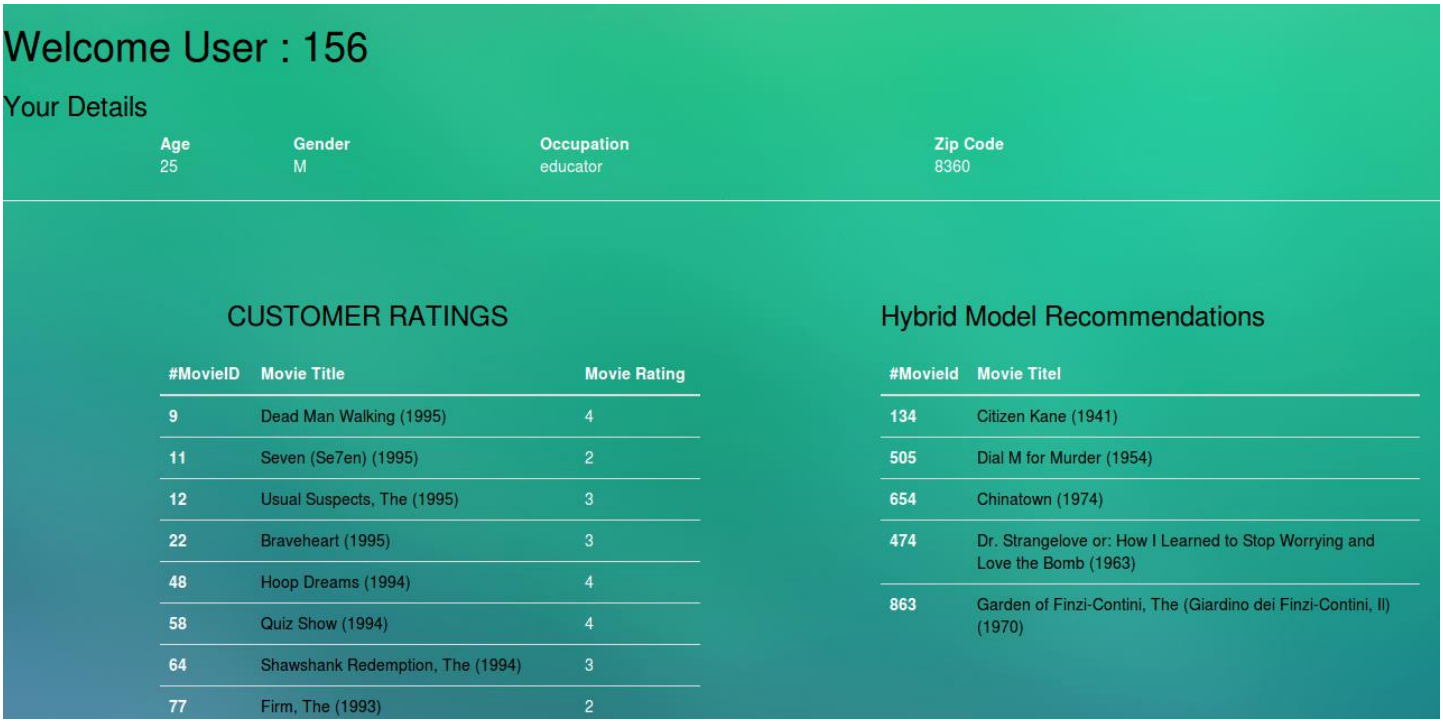
Github Link: https://github.com/Team-HSP/Recommendation-System-Hadoop

Sample Screen-shots from Web UI:

1.



2.



3.

83	Much Ado About Nothing (1993)	3
86	Remains of the Day, The (1993)	4
100	Fargo (1996)	4
124	Lone Star (1996)	3
137	Big Night (1996)	4
157	Platoon (1986)	4
178	12 Angry Men (1957)	5
180	Apocalypse Now (1979)	5
187	Godfather: Part II, The (1974)	5
192	Raging Bull (1980)	4
197	Graduate, The (1967)	5
205	Patton (1970)	3
211	M*A*S*H (1970)	4
276	Leaving Las Vegas (1995)	3
317	In the Name of the Father (1993)	4
318	Schindler's List (1993)	4
346	Jackie Brown (1997)	3

Collaborative Model Recommendations

#MovieId	Movie Titel
987	Underworld (1997)
296	Promesse, La (1996)
437	Amityville 1992: It's About Time (1992)
677	Fire on the Mountain (1996)
857	Paris Was a Woman (1995)

ContentBased Model Recommendations

#MovieId	Movie Titel
593	Stalingrad (1993)
556	Wild Bill (1995)
922	Dead Man (1995)
589	Wild Bunch, The (1969)
470	Tombstone (1993)

4.

Item Details

Movie ID :

257

Movie Title :

Men in Black (1997)

IMBD Url :

http://us.imdb.com/M/title-exact?Men+in+Black+(1997)

Release Date :

04-Jul-1997

Genres :

Action
Adventure
Comedy
Sci-Fi

----------End----------