Recommender Systems using KNN Algorithm

Sai Madhu Bhargav Pallem
Department of Computer Science
University of North Texas
Tel: +1 (940) 208 8229
smbhargavpallem@my.unt.edu

Neha Kasala
Department of Computer Science
University of North Texas
Tel: +1 (832) 710 0707
nehakasala@my.unt.edu

ABSTRACT

In this project, we have implemented k nearest neighbor algorithm in java. For this, we have considered the Movie Lens 10 Million datasets. This dataset contains information about 10M ratings belonging to nearly 8000 users who have rated among 3000 movies. This data set is first stored in a HashMap and these HashMaps are used to calculate the distance between our user and all the remaining users in the data set. After finding the distance, we will select the top k (20) users i.e. the users with less distance from our user will be selected. Based on these 20 users we select the movies that these 20 people have viewed and eliminating the movies that our user viewed. We can make recommendations to our users. The distance here is calculated using the ratings, and Euclidean distance metric is used to calculate the distance. So, we have applied collaborative filtering on the data available. An interface is designed where the user will be provided with entering a user id. After entering the userid, the movie recommendations for that entered user will be shown in a table below that text box. These movies can be clicked and upon clicking, the default browser will be opened redirecting the browser to the google search results of that movie. Where we can find all the related information regarding that movie.

Keywords

Euclidean distance, K-Nearest Neighbor, Recommender System, Collaborative filtering.

1. INTRODUCTION

This project is developed to recommend videos to the users. This recommender system is based on collaborative filtering method. Here the filtering is based on user's behavior and not on the content. The ratings given by the user play a major role here. The similarity between users is based on their behavior towards different videos.

1.1 DATASET

It is a Stable benchmark dataset. It contains 10 million ratings and 100,000 applied to 10,000 movies by 72,000 users. Released on 1/2009.Users were selected at random for inclusion. All users selected had rated at least 20 movies. Unlike previous MovieLens data sets, no demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in two files, movies.dat and ratings.dat. The two data files are encoded in UTF-8 format.

url: *http://grouplens.org/datasets/movielens/10m/*

Rating.dat:

All ratings are contained in the file ratings.dat. Each line of this file represents one rating of one movie by one user, and has the following format:

UserID::MovieID::Rating::Timestamp

The lines within this file are ordered first by UserID, then, within user, by MovieID. Ratings are made on a 5-star scale, with half-star increments.

Movie.dat:

Movie information is contained in the file movies.dat. Each line of this file represents one movie, and has the following format:

MovieID::Title::Genres

MovieID is the real MovieLens id.

1.2 PREPROCESSING

The data we used is preprocessed using java language. The two different files movie.dat and ratings.dat are combined to a whole new data file giving a whole idea of users and their ratings to movie. Altogether, they add up to 10Million dataset.

The Vector Space Matrix C(m x n) is used to express to the preference about the movie n of the user m. Each

cell in the matrix is used to know the users behavior to find an exact match to the active user.

The basic idea in similarity computation between set of items rated by both users (u1 and u2) is based on *Euclidean distance* between two users. The similarity between two recommendation arrays (i.e. person × movie ↦ score) is calculated by.

$$d(x,y) = \sqrt{\sum_{i}^{n}(x_1 - x_2)^2}$$

Euclidean distance is the square root of the sum of squared differences between corresponding elements of the two vectors. Note that the formula treats the values of x and y seriously: no adjustment is made for differences in scale. Euclidean distance is only appropriate for data measured on the same scale.

At a time, we calculate Euclidean distance between one user to all the other users in the data set. The entire preprocessing is done in the function named ***initialize()***. This function will take the two data files and create a model from these two. Creates a hashmap of hashmaps for the ratings file and creates a hashmaps for movies file. Hashing the movie id with movie name and year. This hash map will then be used to calculate the distance of a user from another user. The following is the sample of how data is stored into hashmaps in the project.

```
Scanner scan = new Scanner(new
FileReader("./src/recommender/movies.dat"));
        while(scan.hasNextLine())
        {
            String[] movie =
scan.nextLine().split("::");
            movies.put(Integer.parseInt(movie[0]),
movie[1]);
        }


        scan = new Scanner(new
FileReader("./src/recommender/ratings.dat"));
        while(scan.hasNextLine())
        {
            String[] rating =
scan.nextLine().split("::");
            int user = Integer.parseInt(rating[0]);
            int movie = Integer.parseInt(rating[1]);
            double mRating =
Double.parseDouble(rating[2]);
            if(ratings.containsKey(user))
            {
                ratings.get(user).put(movie, mRating);
            }
            else
            {
                HashMap<Integer, Double> first = new
HashMap<Integer, Double>();
                first.put(movie, mRating);
                ratings.put(user, first);
            }
            if(movieRating.containsKey(movie))
            {
                movieRating.put(movie,
movieRating.get(movie) + mRating);
                movieViewers.put(movie,
movieViewers.get(movie) + 1);
            }
            else
            {
                movieRating.put(movie, mRating);
                movieViewers.put(movie, 1);
            }
        }
```

1.3Nearest-Neighbor Approach

After finding the Euclidean distance between all the users we calculate Nearest-Neighbor for the user. This is carried out by the ***recommend ()*** function which will take the data that is built in initialize function and uses it to recommend the movies to the user that we entered in the text box. The proposed system uses a nearest- neighbor approach to find a subset of all users that have the most similar preference history as the active user. When this neighborhood is found, the similarity between the active user and a neighbor determines how much the neighbor influences the prediction for the active user. If the similarity is high, the neighbors have high influence on the final prediction. Thus, the final prediction is a weighted combination of the neighbor preferences. This following code shows how the data is used in calculating the distances to different users in the hashmap.

```
   HashMap<Integer, Double> ourUserMovies =
ratings.get(usr);
        for(int i:ratings.keySet())
        {
            if(i == usr)
                continue;
            HashMap<Integer, Double> currUserMovies =
ratings.get(i);
            double distance = 0.0;
            for(int j:ourUserMovies.keySet())
            {
                if(currUserMovies.containsKey(j))
                    distance +=
Math.abs((ourUserMovies.get(j) - currUserMovies.get(j))
* (ourUserMovies.get(j) - currUserMovies.get(j)));
            }
            if(distance == 0.0)
                distance = Double.MAX_VALUE;
            distance *= -1;
            if(nearest.containsKey(distance))
            {
                nearest.get(distance).add(i);
```

```
            }
            else
            {
                ArrayList<Integer> ar = new
ArrayList<Integer>();
                ar.add(i);
                nearest.put(distance, ar);
            }

            //System.out.println(i + "->" +
Math.sqrt(distance));
        }
```

Advantages of using Nearest-Neighbor approach: (i) Robust to noisy training data. (ii) Effective if training data is large.

Disadvantages of using Nearest-Neighbor approach: (ii) Need to determine value of parameter K. (ii) Computation cost is very high.

2. INTERFACE

The entire interface of the project is done using *jFrames*. This toolkit is used to develop a text box input which will take the user id as input. A Recommend movies button is provided. User needs to enter the id.

When the user enters his id, the program will read the id as input and starts processing. It calculates Euclidean distances between the active user and other users and applies K-Nearest Neighbor Approach. In our project, we have selected K as 20. Now the nearest 20 users are selected and the movies rated by them are filtered. The program recommends the active user with the movies he didn't watch yet. So, when he enters the id, he is directed to a page with list of all the recommended movie links. The model is built even before the interface shows up on the screen. It takes around 30-40 seconds for the user to be presented with an interface in the meantime, the initialize function will build the hashmaps form the data files.

User can now select his choice and then click the movie link. Now Google Chrome movie link is generated in the backend. On clicking the link, he is directed to the Google Chrome page were full length movie is available. The output screens and their description is discussed in the following section.

3. IMPLEMENTATION

3.1 Requirements

The requirements to run the project are as follows, these requirements are mandatory to run the project:

Operating System: Any Operating System.

Space Required: 250 MB of HDD space is required to maintain the dataset files that are downloaded from the web.

RAM: 50 MB of RAM is required for the faster performance results.

Network: No working internet connection is required to implement and recommend movies, but to watch the recommended movies Internet Connection is required.

3.2 Specifications (Recommended)

The following shows the Specifications of the PC that I've used to evaluate and test the results on. All the results discussed in this paper are completely based on the results that are noted from the system with following specifications.

Operating System: Windows 10 Home Operating System.

RAM: 8 GB

Processor: Intel Core i7 – 5500U CPU @ 2.40 GHz

Network: High Speed Internet connection at 60 Mbps

Data: 10 Million files belonging to the Movie Lens are downloaded and are used as the data source for this project.

3.3 Implementation method

First, the preprocessed data which contains details about the user and the ratings provided by him for each movie is obtained. It is then transformed and stored in a dictionary Data structure.

Now K- Nearest Approach is performed for each user to left over users in the database. The approach is applied to all the users (almost 71000) users in the database. The similarity measure between the users is calculated using Euclidean distance between active user and the existing other user. We have selected k as 15 and performed this approach. For a user, we sort all other users based on the Euclidean distance. Now, the very nearest 15 users from the active users are selected and the left-over users are deleted the similarity is very less and their decision wouldn't help the active user.

After selecting the similar 15 users for an active user, our recommender system plays its major role. It checks all the movies the other users have rated. It selects movies with high ratings which the active user has not yet watched and recommends them to the active user. This Recommender System gives best

recommendations to a user, as it is solely depended on the behavior of the user.

4. CHALLENGES

Storing the Database: For this project, lot of data is needed for processing. We need 250 MB of data to be stored in the hard disk and out of which 50 MB of data is retrieved and placed in RAM to calculate the Euclidean Distances. This leads to high computational cost and needs huge amount of memory. As the data is very high, we need to store all the data on the disk which will take most of the space if hosted on a server. So, we designed a standalone software which can store all the data on the local hard drive and the distances can be calculated whenever required.

We also tried to implement the same using the location of the user, like if a user is new to the database and is registered newly, he/she will not have enough movies that they have reviewed for us to make a recommendation. But, the dataset that we took do not contain the information regarding the user, so we conducted this another dataset that we got.

5. EXPERIMENTAL RESULTS

Before performing the KNN algorithm, the main issue is to find an optimal value for k. If we take very small value for k, all the results may not show up. If the value of k is too high, then the data will overfit and it will show all the movies as recommendations. So, we fed this data to weka and tested for different values of k and finally considered the value of k, where we got least error. The following are the results from weka for various values of k and performing cross validation by splitting the data into 2 parts one for training and one for testing.

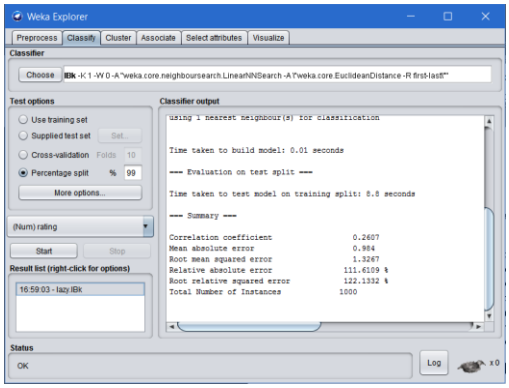

Figure 1: k = 1

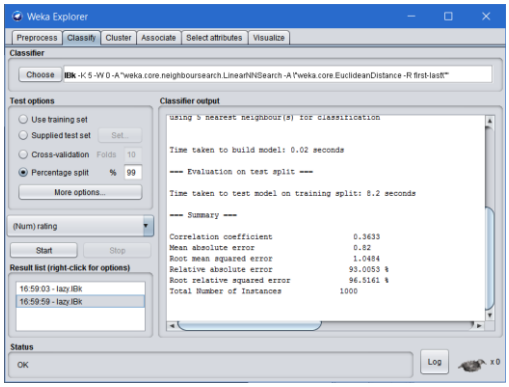

Figure 2: k =5

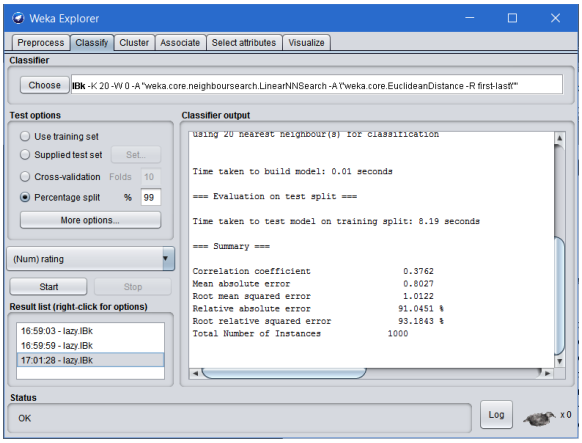

Figure 3: k = 20

K = 0

Relative absolute error 111.6109 %

Root relative squared error 122.1332 %

K = 5

Relative absolute error 93.0053 %

Root relative squared error 96.5161 %

K = 20

Relative absolute error 91.0451 %

Root relative squared error 93.1843 %

Based on this, the values for Relative absolute error and Root relative squared error are lowest at k = 20. So, we picked the value for **k as 20** and continued designing the rest of the project.

The following table shows the sample of how the Movies.dat file looks like. It contains the movie id, movie name and genre.

1::Toy Story (1995):: Adventure | Animation | Children | Comedy | Fantasy

2::Jumanji (1995)::Adventure|Children|Fantasy

3::Grumpier Old Men (1995)::Comedy|Romance

4::Waiting to Exhale (1995)::Comedy|Drama|Romance

5::Father of the Bride Part II (1995)::Comedy

6::Heat (1995)::Action|Crime|Thriller

7::Sabrina (1995)::Comedy|Romance

8::Tom and Huck (1995)::Adventure|Children

9::Sudden Death (1995)::Action

10::GoldenEye (1995)::Action|Adventure|Thriller

The following table data is an example of how the data in ratings.dat file looks like. It contains the userid, movieid, rating to that movie from that user and the time stamp.

1::370::5::838984596

1::377::5::838983834

1::466::5::838984679

1::480::5::838983653

1::616::5::838984941

2::110::5::868245777

2::151::3::868246450

2::260::5::868244562

2::376::3::868245920

2::539::3::868246262

The following are the outputs for the data set on a GUI. The output screens contain the information regarding the recommendations made to the user that we enter in the text box.

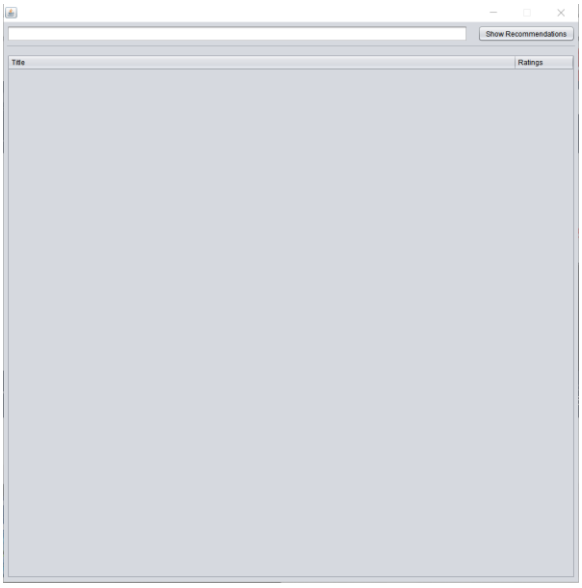

Figure 4: This is how the GUI looks like.

Upon entering the input, the table below is populated by the recommendations that are made to that user.

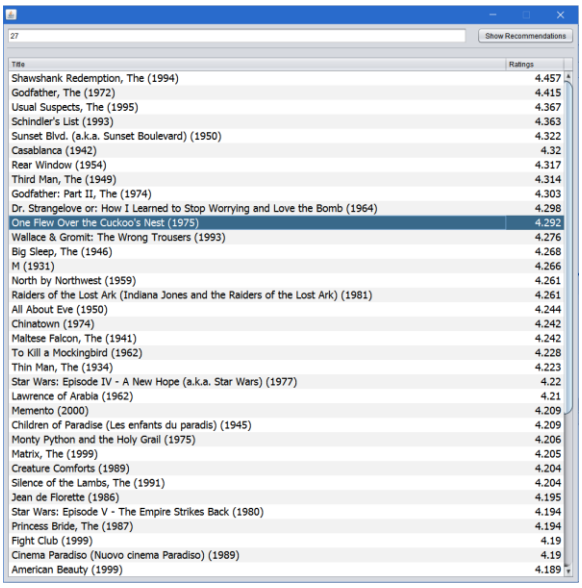

Figure 5: Recommendations for User ID 27

These recommendations that are shown here are sorted per the average rating that all the users gave to those movies. there is also a row sorter included in the columns. You can click the column name to sort

the table accordingly. Only top 50 movies are shown as the recommendations.

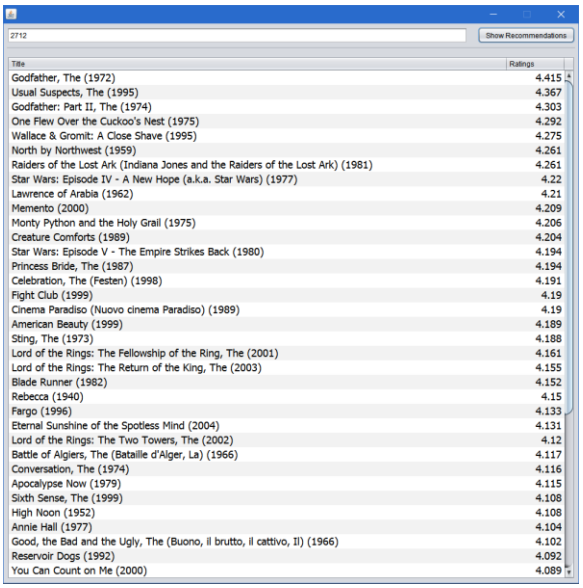

Figure 6: Recommendations for User ID 2712

The following image shows what happens when a user ID that is not present in the data is entered.

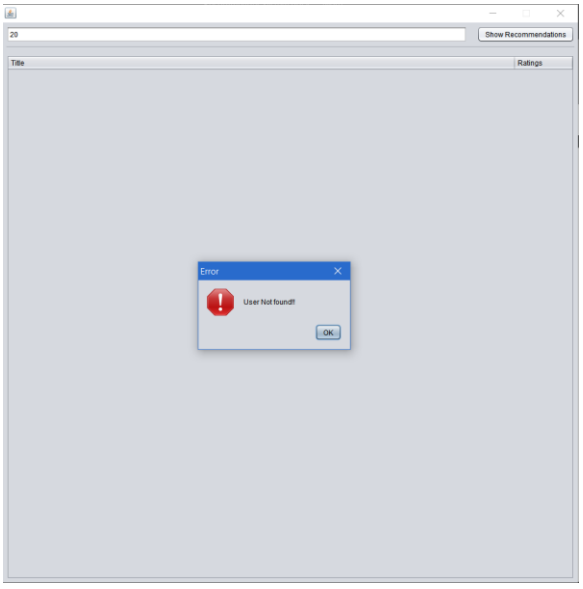

Figure 7: User ID 20 is not present in the database, so it shows an error

In the dataset, all the user ID are not present. So, an error occurs when there are no entries for that userid are present in the hash map. So, to handle such issues, we have handled such situations by showing an error message like this.

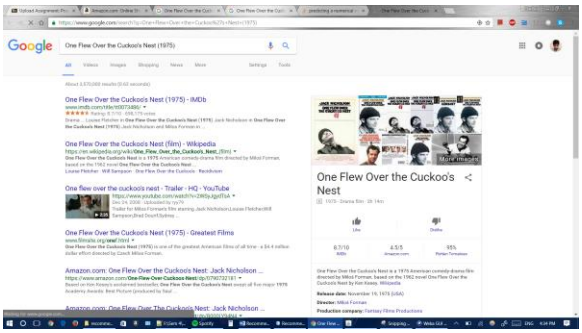

Figure 8: Upon clicking the movie name in the table, System default browser is opened with a search dedicated to the movie we clicked on.

This shows the browser opening the movie search related to the movie in default browser (Chrome).

6. CONCLUSIONS AND FUTURE WORK

We have implemented the KNN algorithm on the movielens data set. The algorithm computes the distance between one user and all the remaining users. The k users with least distance are selected and the movies that the user did not see are selected are the recommendations to that user. This whole project is done on java. For cross validation and for extracting the best value of k, we used Weka.

7. REFERENCES

[1] K Nearest Algorithm. *http://www.cse.unt.edu/~ccaragea/cse5215/lectures/lecture10-knn.pdf*

[2] Information regarding hashmaps and file handling: *https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html*

[3] Weka *http://www.cs.waikato.ac.nz/ml/weka/*

[4] Movie lens 10M dataset that we used from group lens dataset *http://grouplens.org/datasets/movielens/10m/*