Recommendation Systems

Pulkit Gopalani 180564

Achint Agrawal 180028

Abstract

Recommendation systems are used by many platforms to suggest items to their users. The present implementation is an attempt at recommending songs. The recommendations are based on collaborative filtering using cosine similarity as well as Pearson Correlation Coefficient (Using SVD Matrix completion method). The first method is more of a naive approach, calculating similarities of all users, while the second one relies on Matrix factorization based on SVD. The dataset used for the project is the Million Song Dataset, from Kaggle.com .

GitHub link for the project : https://github.com/Pulkit-Go/PT-Project

1. Background

1.1. Introduction to Recommender systems

Recommender systems are being used by many product based platforms like Amazon, Netflix, Spotify etc where user needs to be "recommended" some items, as the total number of items is too large to be explored fully by the user. Out of billions of songs and products, the most relevant are suggested to the user based on his past preferences, and his similarity to other users based on these past preferences. This is based on the basic assumption that if 2 people have listened to similar songs in the past, they will also like similar songs in the future.

These platforms use sophisticated ML based algorithms like K-Nearest neighbors, Clustering etc. and a mixture of Collaborative filtering based and content based recommendations.

1.2. Similarity measures

Similarity measures are basically the different type of methods used to calculate how similar two users or items are, based on the difference between the specific ratings. Two measures are quite popular - Cosine based similarity and Pearson Correlation Coefficient. One major difference between the two is that cosine similarity is based on raw data, whereas Pearson correlation coefficient is more of a "normalized" measure, meaning that the raw data is not directly used, instead the standardized data (subtracting the mean from each entry of a given row/column) is used. Therefore it can (and does) lead to negative correlation, which signifies that the two objects are far from being similar.

1.3. Evaluation Metrics

The evaluation is an important part of the recommender system design, telling us how accurate the prediction model actually is. The evaluation is done using MAP (Mean Average Precision) score, in which some of the songs listened by the user are masked, and based on the remaining history, the predictions and the hidden songs are compared. More the matching songs, better the MAP score. Basically,

$$MAP\ score \quad = \quad \frac{Number\ of\ matching\ predictions}{Total\ number\ of\ hidden\ songs}$$

2. Problem Overview and Approach to solution

2.1. Dataset used for recommendations

The data used for calculating similarities, making recommendations and evaluation is the Million Song Dataset from Kaggle, which has the data of a million users and their usage history. In the present project we have used the kaggle-song-evaluation-triplets file which has the User ID of each user, corresponding to which the songs which have been listened to by the user have been listed, along with the number of times the specific song has been listened.

One issue with such kind of dataset is that people don't listen to as many songs as there are available. The average user has listened to only 30-35 songs, while the data is available for thousands of songs. This leads to a sparse data matrix, with most of the elements empty (the matrix in question here has a sparsity $>$0.95, i.e. in a 10 x 10 sub-matrix of the matrix, there are high chances that only 5-6

elements are actually filled). There are many users who have not listened to many songs (< 5). So to address these issues, the data has been filtered out to only the top 2000 users with the most number of songs listened. The matrix is then effectively converted to a denser matrix with comparatively much more entries per user.

2.2. Calculating similarities

2.2.1. Cosine based similarity

Cosine similarity is based on the property of vectors that the cosine of angle between them is the dot product of the normalized vectors (unit norm vectors pointing along the original vectors). More specifically,

$$cos\ \theta \quad = \quad \frac{\mathbf{x} \cdot \mathbf{y}}{\| \mathbf{x} \| \| \mathbf{y} \|}$$

This means the similarity will lie between -1 and 1. For this particular case, the ratings are non-negative, so the cosine based similarity will also be non-negative, i.e. in the interval $[0, 1]$. We take the product of a specific user's ratings with that of another user (the vector here has a dimension = number of songs). Similarity therefore can be written as:

$$sim(i,j) \quad = \quad \frac{\sum\limits_{s \in S_{ij}} r_{i,s} * r_{j,s}}{\sqrt{\sum\limits_{s \in S_{ij}} r_{i,s}^2} \sqrt{\sum\limits_{s \in S_{ij}} r_{j,s}^2}}$$

where S_{ij} is the set of songs for which both users have rated it a non-zero number. $r_{i,s}$ and $r_{j,s}$ is the rating given by the i-th and j-th user.

2.2.2. Pearson Correlation Coefficient

Pearson correlation is very much similar to cosine based similarity, differing only in the fact that it considers the standardized or mean-relative ratings for calculating correlations. In this case the correlation coefficients can be both positive or negative based on whether the items being compared are alike or different.

Therefore mathematically PCC can be written as

$$corr(i,j) \quad = \quad \frac{\sum\limits_{s \in S_{ij}} (r_{i,s} - \bar{r_i}) * (r_{j,s} - \bar{r_j})}{\sqrt{\sum\limits_{s \in S_{ij}} (r_{i,s} - \bar{r_i})^2} \sqrt{\sum\limits_{s \in S_{ij}} (r_{j,s} - \bar{r_j})^2}}.$$

The calculation of predicted ratings in this case is done using SVD and matrix completion methods.

2.3. Predicting the ratings and recommending songs

2.3.1. Rating based on user-user similarity

In this part, weights assigned to the users are their similarities with the user in question. As can be seen,

$$r_{c,s} = \frac{\sum_{c' \in C} sim(c, c') * r_{c',s}}{\sum_{c' \in C} sim(c, c')}$$

is the predicted rating for user c and song s. Note that more the similarity between c and c', more the weightage given to c' rating for predicting the rating of c.

Now, using argpartition and argsort functions of numpy library the top-k songs (here k = 50) based on predicted ratings, not listened to earlier were recommended to the user.

2.3.2. Rating based on matrix completion

The TruncateSVD method of scikit-learn library was used to carry out the matrix decomposition in this project. It reduces the dimensionality of data being used and then completes the matrix. Based on this matrix the correlation coefficients of all songs vs themselves were calculated using Numpy's corrcoef() function. The higher the correlation, more similar the songs were. Based on these the highest correlated songs (w.r.t. the song rated best by the given user) were recommended to the user.

3. Implementation and evaluation

The input was read from the file "Kaggle-evaluation-triplets" using .strip and .read functions, used for file handling in python.

3.1. Recommendations based on user-user similarity

The program was run for 100 and 700 users subsequently, and the snapshots are appended below.

As can be observed, the time taken by this method is too large for higher number of users, a problem which is solved to some extent in the next implementation.

3.2. Recommendations based on song-song similarity (SVD matrix completion)

This implementation is faster than the above implementation, taking almost one-fourth of the time for 100 users than in the first case. As in the previous case the numbers indicate the song indices.

This program too was run for 100 and 700 users subsequently. The output snapshots are appended below.

As can be seen, this method is much faster than the previous one in case of recommending songs.

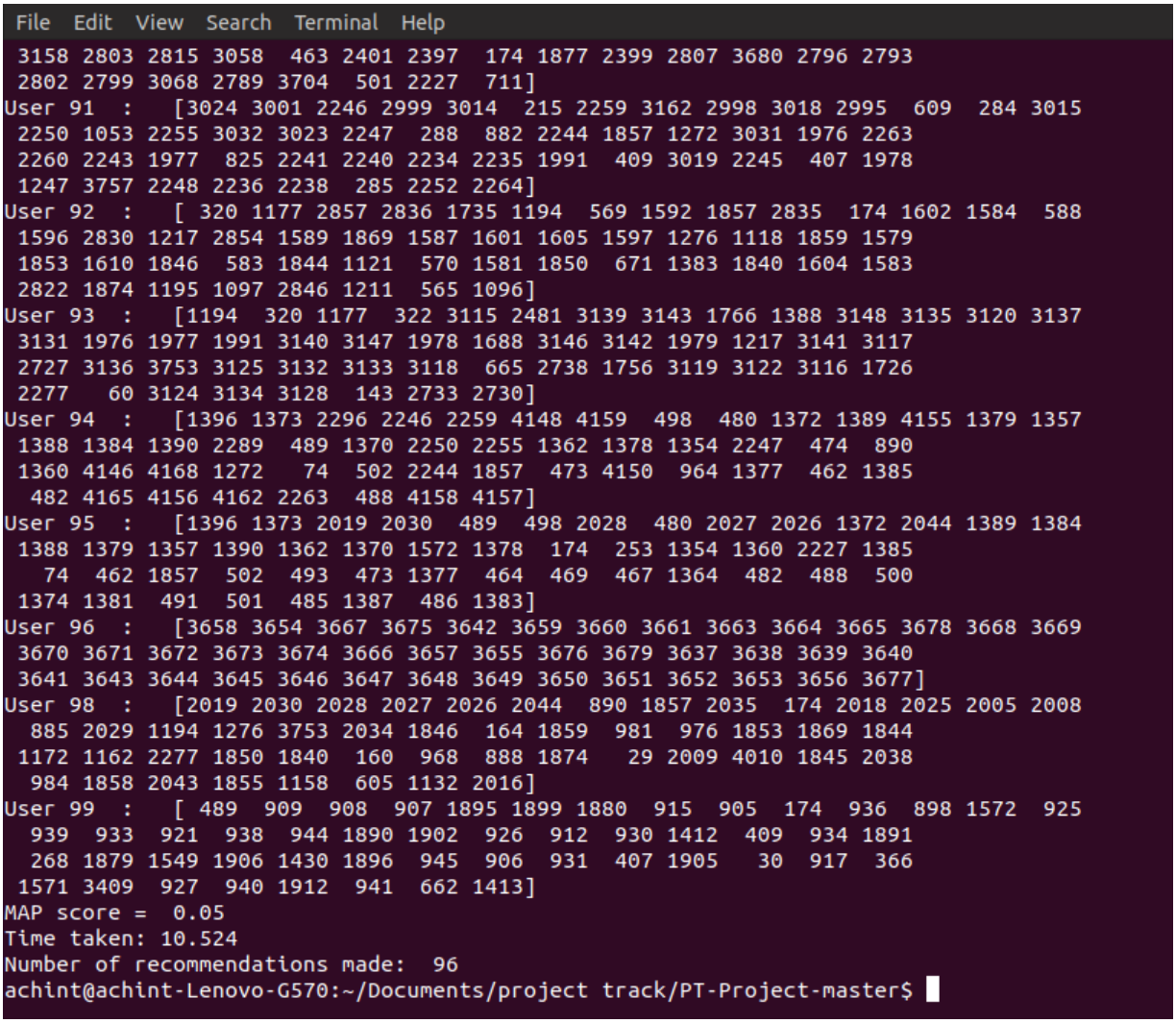

Figure 1: 100 users, first approach

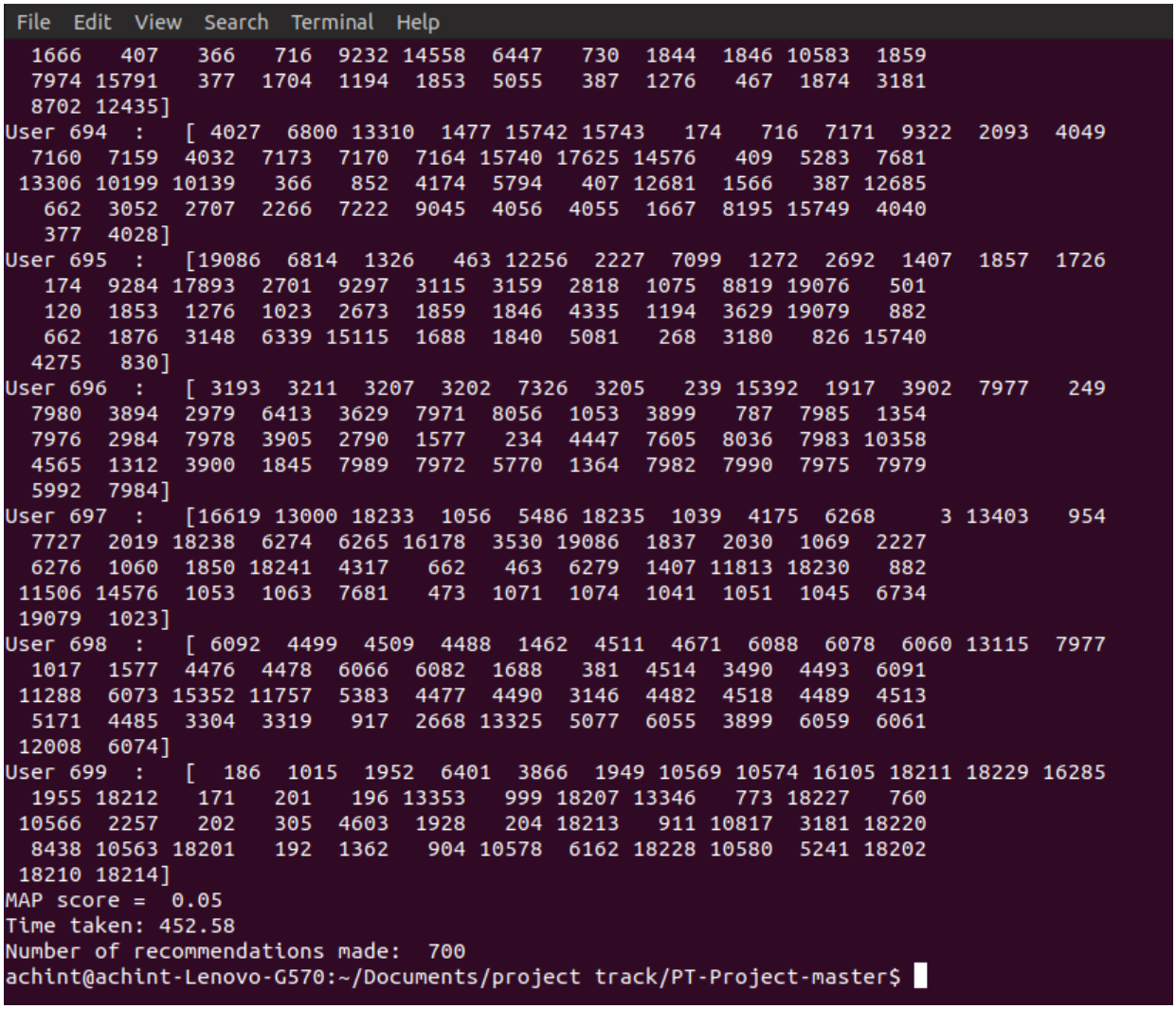

Figure 2: 700 users, first approach

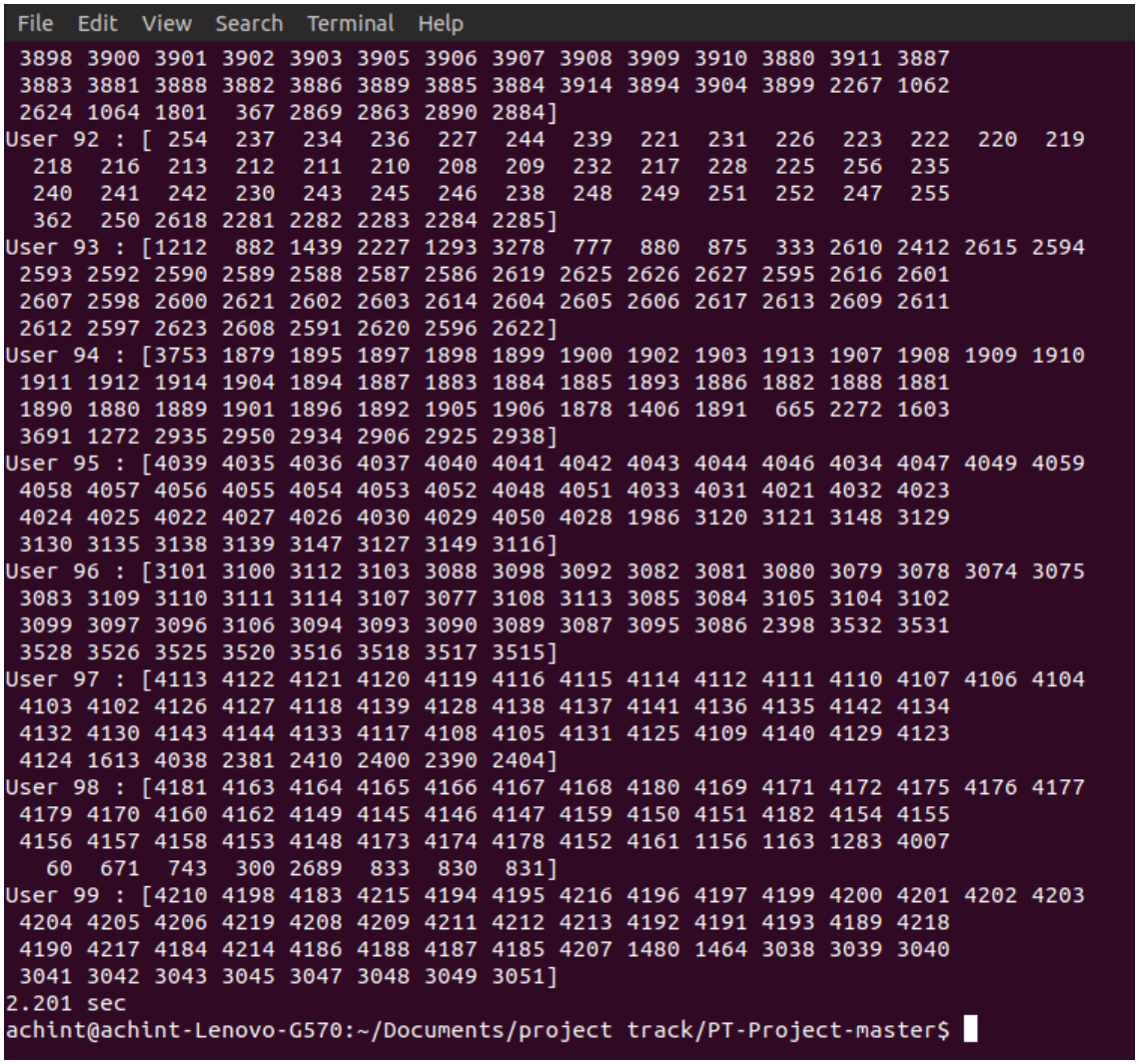

Figure 3: 100 users, second approach

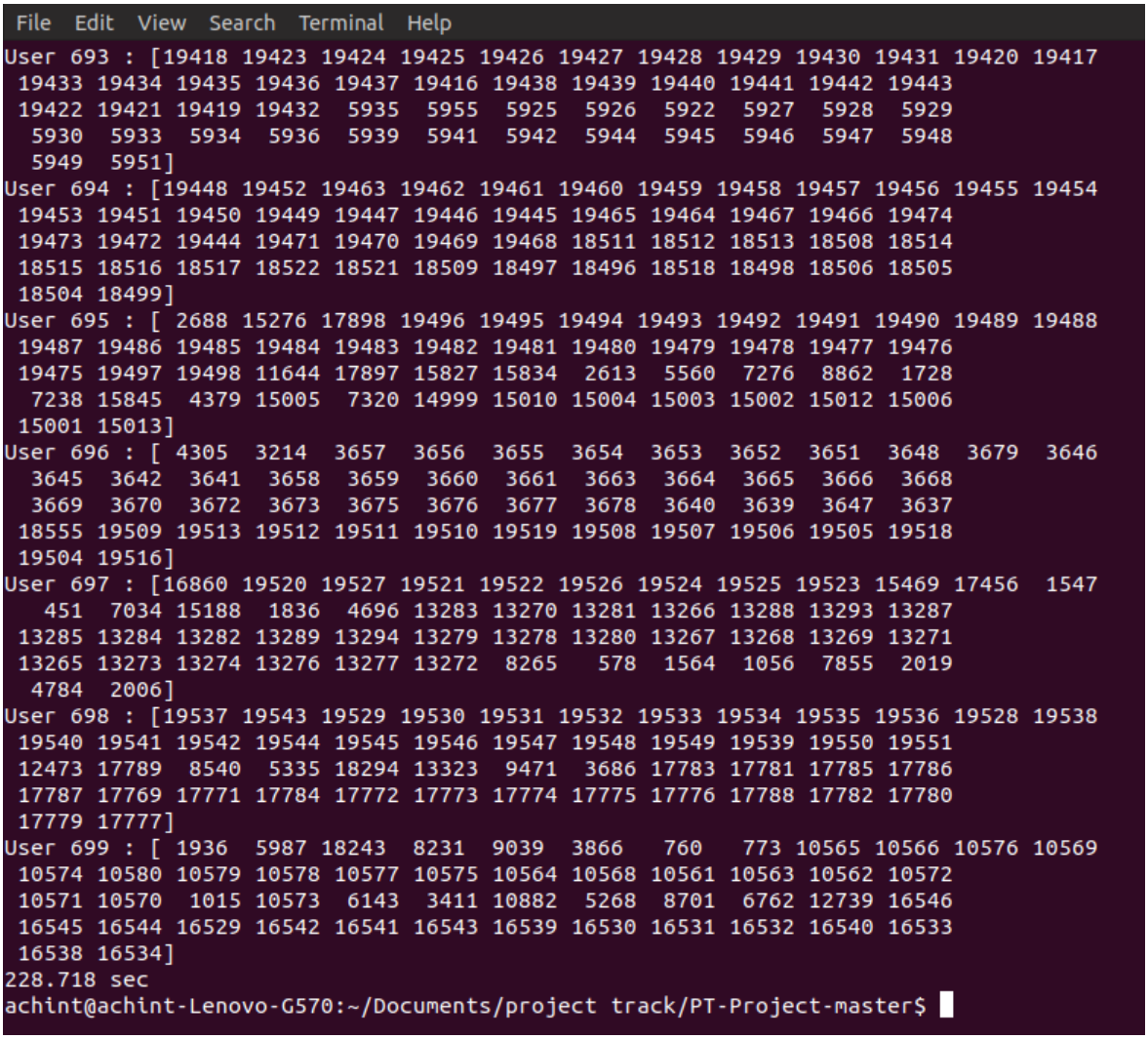

Figure 4: 700 users, second approach