Project 1

Team Members: Domitille Chambon, Steven Hobson

An RDD (Resilient Distributed Dataset) is used to split up the data among clusters. The data is separated and is then brought together after doing the calculations. These calculations are done in parallel on the cluster so that it is faster run times than doing it all in memory from the local computer. It is primarily used in data applications that have millions of rows of data because it is so much faster.

Features used for Recommendation

The recommended songs are songs that the user has not listened to and are liked by others that are higher than the user's average liked song. We also used the cosine similarity to find similar users compared to the selected user and recommended songs based on the top songs of the similar users by using song weights.

Flow of the Code

Step 1

We used the already built code to tell our computer where to find Apache Spark. We also read both the triplet file and Kaggle songs files into RDDs. We mapped the triplet into lists so that we could join them into the updatedTriplet RDD. We subtract the total number of songs with the distinct count of the updatedTriplet (songs that inner joined which means they have a rating) to get the number of songs without a rating.

Step 2

We make the updatedTriplet into a dictionary with lists of lists and grouped by user. We then created a sumPlayCount that gets applied to each value in the dictionary and add as a column. The rating is then calculated by dividing the # played by the sumPlay count to get a rating and round by 4 for each song. We get rid of play count and sumPlayCount and we are left with `[('e4332e11f4df6dd26673bb6b085e9a2bbdc9b8a5', [['25150', 0.037],…]])]` which is our user, song index, and song rating.

Step 3

We pick one selectedUser where we average their ratings as a threshold number. We also got the selectedUsers top song with the highest rating. We then pulled all the users who have listened to the user's top song and whose rating is higher than our user. We created a temporary list without the top song as potential songs and removed songs that were less than the threshold. We took it a step farther and removed songs from user songs that the user has already listened to. We then sorted the recommendation list based on the highest ratings.

Step 4

We first take the RDD with the User and take only 275 songs from the users. The unique songs is a unique list of songs that have been listened to from those 275 users. We then calculate the cosine matrix. The userSimilarities is the cosine matrix that gets calculated. We get similar users by using lists and appending to a list. We then removed pairs that have cosine similarities equal to 1 and 0 because

those are users that have not listened to the same songs. We sorted the pairs from highest similarity score to lowest and removed duplicate pairs to get the top 5 most similar users. We then weighted the songs based on how many users like those songs that were similar. We then picked the top 5 song recommendations based on the weights and give those to our selected user.