Omar Rizk
NetID: or125

Distributed Computing with Spark: Analyzing Reddit and Netflix Data

1) For each program, describe what transformations you did to get to the result.

RedditPhotoImpact: I set up a JavaRDD so that I could store all the data read in the CSV file. I then did a mapToPair operation which maps the JavaRDD onto a JavaPairRDD. Each individual line was mapped to a tuple that had a key-value pair which is type <Integer, Integer> where the image ID is the key and the impact is the value. Right after this I used the reduceByKey transformation so that I can garner the final key-value pair which is then sorted to print the results.

RedditHourImpact: Similar to RedditPhotoImpact, I read and parsed the lines that were to be stored in the JavaRDD. I then converted the Unix time to EST timezone. At least locally, this offset the results by one hour to account for a different timezone than was set for the system, which may or may not apply to the Spark node as well. I then used mapToPair on the lines which were mapped to JavaPairRDD. It has the key-value pair which is type <Integer, Integer> where hours is the key and impact is the value. I then used the reduceByKey transformation to gather the output and sort it in ascending order.

NetflixMovieAverage: Similar to the other programs, I again read and parsed the lines to be stored in the JavaRDD. I applied the mapToPair operation from the lines to a JavaPairRDD with the key-value pair type <Integer, Tuple2<Double, Integer>>. The key is the movie ID and the value is a tuple with the rating and the count. After this I use reduceByKey transformation to add the total ratings for each individual movie and ratings. I then map the resulting key-value pairs to a tuple with the use of mapToPair once again. I then gather results that are sorted by impacts ascending and output the result.

NetflixGraphGenerate: I again read and parsed the lines to be stored in the JavaRDD. MapToPair was used where the JavaPairRDD key-value pair type is <Tuple2<Integer, Integer>, Integer>. The key is a tuple with movie ID, rating, and the value is the corresponding customer ID. I then emulate a self-join on matching keys by performing a cartesian product. I then make sure there are no duplicate counts by removing any entries that are structured as <x,x>, <x,y> when x<y. After this occurs the result is a key with the movie ID and rating, and a value of a tuple customer ID where the pair has the same

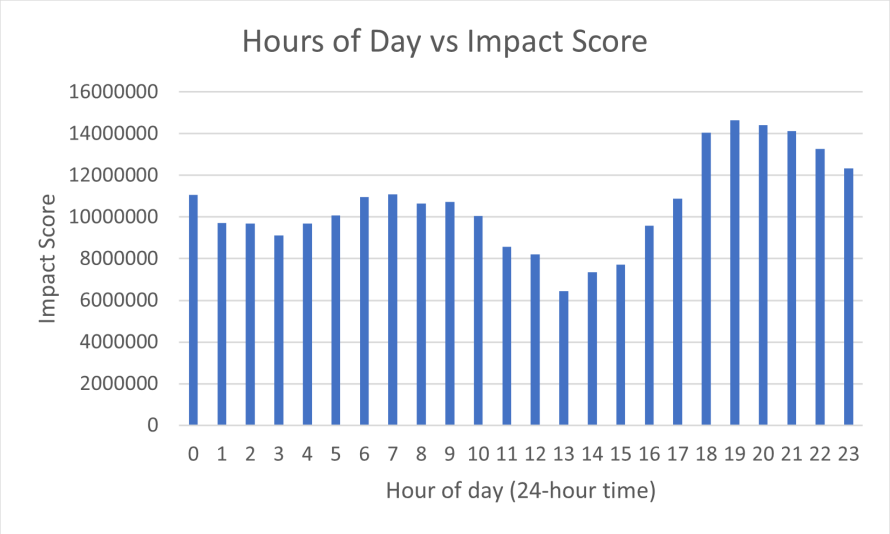
rating for the same movie. I then use mapToPair to map customers by commonality. I then use reduceByKey to add up all counts by commonality and output the results.

2) Based on the result from RedditPhotoImpact, what was the most impactful photo for the whole data set?

The most impactful photo was image ID 1437 at 192896 total impact score.

3) Based on the result from RedditHourImpact, what hours of the day (in EST) did submitted posts have the most impact/reach?

The greatest impact is on posts in the evening, around 6pm- midnight EST. I have compiled an excel chart of the output generated by RedditHourImpact.



4) Based on the result from NetflixMovieAverage, name 2-3 movies that had the highest average rating.

The highest 3 movies by average rating are ID 14961 at 4.72, ID 7230 at 4.72, and ID 7057 at 4.7.

Reflection

I definitely had difficulty with this project especially in the beginning when I was trying to set up everything and cut past all of the errors I was getting. I ended up installing Spark locally on a personal linux machine to run the code. The first three programs were manageable, but I was very confused with NetflixGraphGenerate. I managed to use a

cartesian product to emulate a SQL join so that I could determine the unique pairs of customers who have the same rating for the same movie. I'm not sure what else I would have done to join the different reviews on each of the movies by customer ID. All in all, the project was manageable and going through the Spark lecture notes helped me better understand the concepts that are implemented here.