Parth Shah
04/10/21

Project 2: RU-Spark

First I downloaded the Project-Template package provided in project requirements, then I imported the package into the eclipse IDE, as described in the instruction I first read about Apache Spark from resources provided in week 8's module. After reading about spark and it's different transformations, I looked at the WordCount class provided as reference to furthermore solidified knowledge about those transformations. After a few days of reviewing those information I started writing code for first class which was RedditPhotoImpact.java, main objective of this class was to calculate the image impact on a particular post, to do this we will consider data such as upvote, downvote and comments on that post. The class has already created a String that contains a path to the dataset which is a csv file for reddit data. For start off in the main method I created a SparkSession which accepts the appName. After that I created a photoImpact method of void type that accepts a String input path and spark as an argument. I used spark.read() method to read the content of the csv file and then converted it to JavaRDD of rows. One of the spark transformations used in the class was mapToPair function, it created a JavaPairRDD, so each element inside the RDD contained a key and value. I set the key for the RDD to be image id which was the 1st row in the file provided, Since we are interested in upvote, downvote and comments I parsed them into a long and summed them to get a total impact score. I returned a tuple which had a String imageId as key and Long Impact score as value. Now that I had different key values, I used another spark transformation reduceByKey() function to get the sum of all the posts that have the same key. reduceByKey traverses around all the imageId keys and sums the corresponding value into a new pairRDD, which means that after

the function completes executing I will have one pairRDD for each imageId. After that I collected the pairRDDs converted into a list of tuples of String and Long. I used a for loop to traverse around the list of tuples and print out the tuple key which is imageid and tuple value which is photoimpact score. Once the code was completed for this class I ran maven build for the entire project to get the jar file for the class, to run the jar file I used ilab. After successfully uploading the jar file onto the ilab, I was successfully able to run the code and store the output in the output folder of ilab machines. The most impactful photo among all the photos was a photo with image id 1437 and the impact score was 192896. I got the result by going to terminal and sorting the output text for large reddit file, to sort the file I typed in first sort -k2n which sorts the data is ascending order and was able to get this result by looking at the last data which is going to be biggest if we sorted in ascending order.

Second class I created was RedditHourImpact where we have to find the most ideal time to repost on reddit. Similar to the photoimpact class I created spark session and sent the dataset and spark session to the method which I created. Again in the method I first used spark.read() functionality to read the csv file and convert it to javaRDD which reads every row of the dataset. I created an Instant object that uses a second column which is a timestamp and parses it to a long object and converts it to instant of time which is a Java object that deals with time stamps. Since it was mentioned in the description I defined the zone to America/NewYork. After that I created a ZoneDateTime instant by passing the instant and the zone. It returns the instant in the correct zone, so I used the get hour method to get the hour and assigned it to the Integer variable which will tell us what time the post happened. After that I perform the same actions as RedditPhotoImpact to count the number of upvotes, downvotes and comments and some them

and assign it to one long variable. Instead of image id it returns the hour when the post happened as key and same value which is a sum of all upvote,downvote and comments. Then again I reduceByKey() transformation that returns only one tuple per hour and sum of all photo impact scores. Then I collected current RDDs in a list of tuples, and to ensure every single hour has value I created a hashMap which contains hours as key and impact scores as value. Then I used a for loop to traverse through the list of tuples and put in hours and impact scores into the map. It will only contain every hour inside the data set but not all 23 hours, for that I created another for loop checks if every hour from 0 to 23 contains the hour key, and if it doesn't it puts that hour as key and 0 as value which means there were no posts at that hour. After that I printed all the keys which are hour 0 to 23 and value for that key. Again once the code was completed I ran the project template as maven build and created an executable jar file for the class, I exported that into my ilab machine folder and stored the output into the output folder. Looking at the RedditHourImpact dataset the best hour to post is at hour 20 or 8:00 PM EST because it has an impact score of 15057971. I got the result the same way by sorting the text file on the terminal using sort -k2n which sorts the text in ascending order.

After Reddit classes were completed, I read netflix data instructions and started working on the NetflixMovieAverageRating java file. Objective for this file was to calculate the average rating for each movie, as before two classes I first created a spark session and sent a dataset and spark as an argument for the method which I created. Again in the method I used spark.read() to read through the data from the dataset and convert into javaRDD. First transformation in this file I used was mapToPair, key was movie id which was in the first column and rating as a value from column 3. After that I created a tuple of tuples, so the tuple had movieid as key and a tuple of

rating and 1 as value, so it will only have content of one single rating. To count the average I used the transformation reduceByKey which keeps movieid as key and sums the value of each tuple. So it will sum the rating with another tuple with the same movie id, and also sums 1 which keeps the count of how many times. To get the average rating I used a transformation map, which maps every single tuple with image id. It retrieves the key image id and value average, which is the sum of each rating and dividing my number of ratings. Again I collect the rdd's in a list of tuples. To format the decimal point I used decimalformat and set rounding mode to down. I again used the for loop to traverse through the tuple and print out key and value in correct decimal format. I repeated the same process of creating a jar file by maven build and using ilab to get the output for the file. For the netflix movie average rating dataset looking at the large dataset three movies that I will recommend is one with movie id 14961 which's average rating is 4.72, other is movieid 7230 which has average rating 4.71 and 7057 with average rating 4.7. I got the result in the same way by sorting the text file in ascending order and looking at the highest rated.

For the last file of the project we had to create a recommendation graph. I started again by spark session and passing the dataset and spark into the method which I created. spark.read() function to read through the dataset and convert into javaRDD. I used transformation mapToPair by defining a key which is a tuple with movieid and rating, the value customer id. To group all the movieid with the same rating I used transformation groupbykey, I didn't use reducebykey because each group will contain all the customer id that have the same rating for the same move. With transformation flatMap I stacked customer id in group and added them to a list. After that I sorted the customerId in ascending order so lower customerIds are at the beginning. Then I

created edges which are tuples from one customerId to another, we know that each customer had the same rating so there's a relationship between each of the customers in the group. I created edges by creating list of tuples and using for loop to traverse around customerId from first and one before the last customerId, and other for loop I traversed around next customerId to last customerId so that I can create tuple of one customerId to another. Since I used flatMap transformation, I have to return an iterator of edges. So flatMap converts the iterator to a list of tuples which are edges, each edge contains coustomerId1 and customerId2. I used mapToPair to map their pair to a counter to check how many times the customerid tuple repeats which means that how many times two customers have watched the movie and gave the same rating. To get same customer relationship and how many times they rated the movie the same. Each tuple in reduceByKey will represent unique customers and will not repeat the same combination, which corresponds to graph. Once I got the graph I collected it in a list of tuples, and printed customerid1 and 2 in parentheses and the number of times those tuples happened in the graph. Again once it was done I ran the project as a maven build and exported the jar file onto ilab and stored the output in the output folder.

The hardest part about implementing the project was to understand how different spark transformations such as map,flatmap, reduceByKey and groupByKey work over Spark Rdds and how they were useful to solve the problem. To overcome this challenge I first went through the notes and lectures on spark and the resources provided in that module. I also read spark RDD API documentation on "https://spark.apache.org" to understand the transformation and being able to use them to get the final solution, understanding that each time any transformation is applied, spark creates new RDD and thus input RDD cannot be changed since they are

immutable was clinical. One of the other challenges I faced initially was configuring through ilab, first I was not able to login using my cs account but after watching the introduction video provided on the website I was able to login. After that it was a challenge of exporting my local files to ilab machines and then figuring out commands to write on the terminal to generate the output file. After reading through project description and going through the introduction video again it was easier to use ilab machines.

Even though I completed my project on time there were some mistakes I could have avoid, such as I didn't start reading the project description on the first day it got available to us which later came to haunt me as I spent more time then I wanted on going through the project description and understanding the requirement, made last few days before project submission challenging.