Part A. Message Length Analysis

The aim of Part A is to create a Histogram plot that portrays the tweet length distribution of the Twitter data set /data/olympictweets2016rio within Hadoop.

The data is in the format: epoch_time;tweetId;tweet(including #hashtags);device.
(0) – epoch time, (1) – tweet ID, (2) – tweet, (3) – device. As seen, each index is separated by semicolons.

```
String[] tweets = value.toString().split(";");

if (tweets.length == 4 && tweets[2].length() <= 140)
{
tweetLength = tweets[2].length();
```

In the Mapper (TwitterMapper), the line of code above sets up a "tweets" variable as the separated string of text, split by semi colons. After splitting the data via semi colons, conditions need to be set to limit errors within our output. The if statement has the following conditions:

1. The string must have 4 objects (epoch time, tweets id, tweets and device)
2. The length of the 3rd index (2) must be less than or equal to 140 characters as that is the maximum amount of characters a tweet can have at the given time.

Within the if statement, the length of the tweet will be stored as variable. The "2" finds the third index in the tweet array which is the actual tweet.

Now that the length of the tweet has been found, the aggregation into bins of five needs to be computed. The method to do this is to take the length of the tweet and turn it into the nearest multiple of 5 so it can be put in a bin.

```
data.set((tweetLength - (tweetLength - 1) % 5) + 4));

context.write(data, one);
```

The modulo operation returns the remainder of dividing two numbers. In this case, the modulo is taken between the (tweet length – 1) and 5. That result is then taken away from the tweet length in this instance and then 4 is added. This will return a number that will be divisible by 5 which will be the classes for the bins. The last line "context.write", writes the tweet length at that instance and the increment of 1; this will then be sent to the reducer.

```
int sum = 0;
for (IntWritable value : values) {

sum+=value.get();
}
result.set(sum);

context.write(key,result);
```

In the reducer (TwitterReducer), this is where all of the like keys are collected and aggregated. The key being the variable "tweetLength" which holds the length of each tweet.

- Histogram Analysis

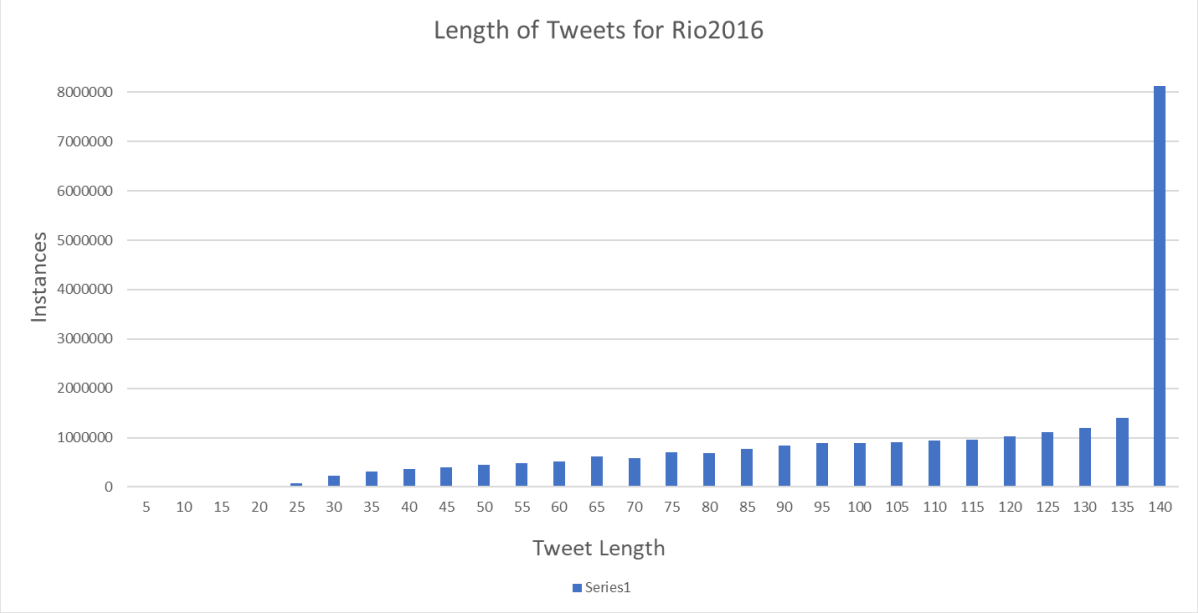

Figure 1 - Rio2016 Histogram

This is the histogram for the length of the tweets for the Rio Olympics 2016.
The bins seen in Figure 1 increase in increments of five from 5 to 140 characters. Characters with length 1-5 will be in the first bin conveyed by the number 5. This follows suit, up until 136 – 140 characters which is depicted by the number 140.

As seen in Figure 1, the maximum tweet length is to be seen as 140 characters. This condition was set in the Mapper. The tweet length with the least amount of instances/number of tweets is the first bin that illustrates all tweets with length 1 to 5. This count is at 0 and can be explained due to the fact that the data on the file in the Hadoop file system collects all data with "#Rio2016" or "#RioOlympics" in the tweet. Because of this, the minimum length of the tweets is likely to be 8 as that is how many characters are in "#Rio2016". From the image, it may portray that the first four bins do not hold any instances, however due to scaling, the bars cannot be seen. The first bin is the only bin that doesn't have any tweets, the other bins hold 11,447, 10,244 and 31,926 tweets respectively. Overall, there is a general increase in the tweet length during the 2016 Rio Olympic games.

- Tweet length of 5 and its result of 0 was not written in the results from the MapReduce job but was included for the purpose of analysis in the histogram.

The interpretation of this visual is affected by the potential use of links within any given tweet. For example, if an image is attached to the tweet, a URL will be provided for that image. According to Twitter, every link included in a tweet will be adjusted to 23 characters, regardless of its original length (Twitter, 2017).

Part B. Time Analysis

1. The aim of PartB.1 is to create a bar chart showing how frequent tweets were in each hour of the day across the event.

In the Mapper (Time Mapper), since we are working with the same input data, the check for four objects still holds.

```
String epoch_time = tweets[0];

if (epoch_time.length() == 13){
long epoch = Long.parseLong(epoch_time);
```

After the check for four objects, the "epoch_time" variable needs to be assigned to the correct index within the data that carries the actual epoch time of the tweets. The epoch time is the number of seconds that have passed since Jan 1st 1970. There seemed to be data in the 0th field that didn't depict the epoch_time of a tweet. To rectify this, an if statement was invoked to check the number of characters in 0th field. Hence why the check for epoch time is set to 13. Following that, since the epoch_time was originally set as a string array, it now needs to be set to as a long variable so manipulation of each epoch time can be made easily. The method to retrieve the hour from the epoch time is as follows:

```
SimpleDateFormat sdf = new SimpleDateFormat("HH");

sdf.setTimeZone(TimeZone.getTimeZone("UTC"));

String hour = sdf.format(epoch);

data.set(Integer.parseInt(hour));
```

SimpleDateFormat is helping to retrieve the hour of the day given the epoch time. As the "sdf" variable is being set, it is key to keep in mind that all tweets around the world are posted in different time zones. The "sdf" variable needs to take into account the time zone. Therefore, "*getTimeZone*" is employed to get the UTC of all epoch times so that they can become comparable. Finally, the epoch time was originally set as a string. However, this needs to be changed to be sent off to the Reducer (TimeReducer), with an increment of one, as an Integer because the desired keys are integers from 0 to 23, hence *"Integer.parseInt(hour)"*. The Reducer will collate all keys appropriately and the result will be the number of tweets that were posted in each hour of the event.

- Bar Plot Analysis

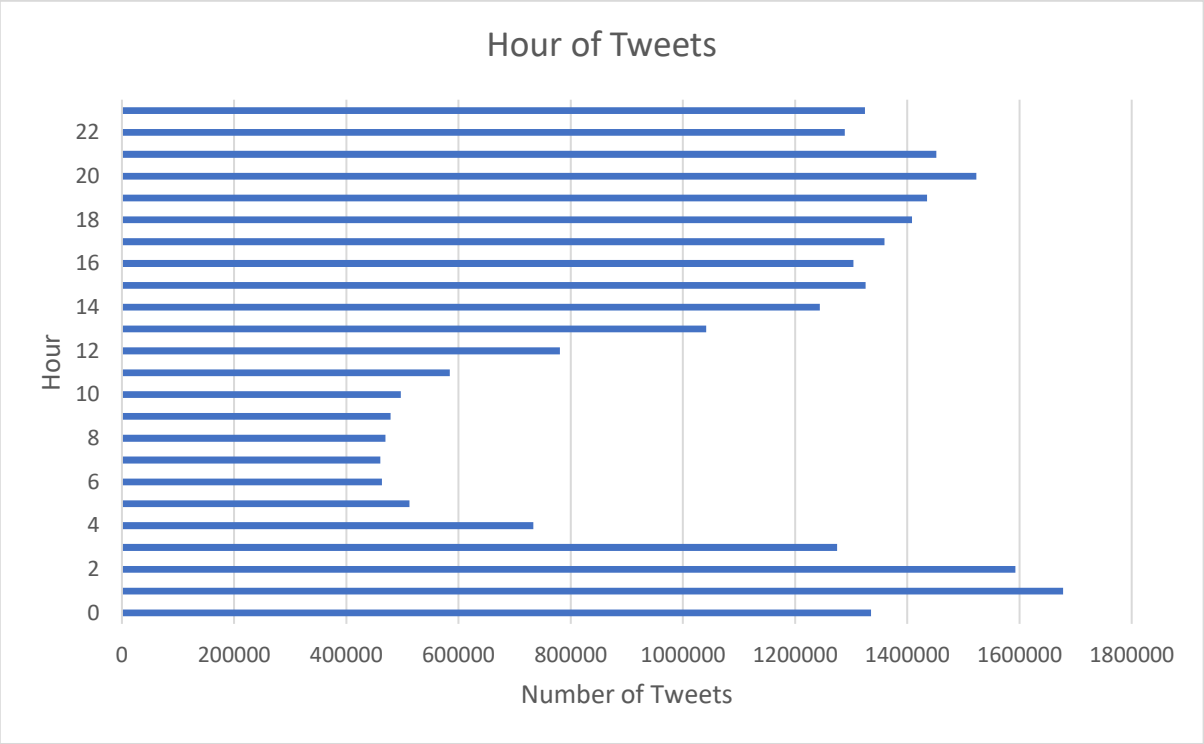

Figure 2 Bar Plot for Hour of tweet

This is the bar plot for the number of tweets in each hour of the 2016 Rio Olympic Games. As seen in Figure 2, the 7th hour, portraying 7am, had the lowest number of tweets at 460,629 tweets (with reference to Appendix 1). 1am showed to have the highest number of tweets within that hour at 1,677,571 tweets. 1am is 11pm local time in Rio de Janerio, where the Games were being held. This makes sense to be the most popular hour as it is likely that the popular events took place at this hour. Take a given day of the event (Sunday 7th August); according to Vulture.com, the Women's 400m freestyle and Men's 4x100m relay took place from 9pm to 11pm local time (Lyons, 2016).

From 4am to 12pm, the activity during the event on Twitter seemed to be relatively less than the remaining hours of the day.

2. The aim of Part B.2 is to compute the top 10 hashtags in the most popular hour according to PartB.1.

The Mapper (HashMapper) for this computation follows suit to PartB.1, the following code is added:

```
if (Integer.parseInt(hour) == 1){
 Pattern p = Pattern.compile("(#\\w+)\\b");
Matcher m = p.matcher(tweets[2].toLowerCase());

while (m.find()){
data.set(m.group());
context.write(data, one);
```

PartB.1 illustrated that 1 am was the most popular time during the Rio games based of the given data set. Therefore, instead of looking to compute the hour, it is hard coded in the if statement. The use of Regular Expression (regex) in Java is needed to extract each hashtag within each tweet in the given hour of 1am. Pattern and Matcher are classes within Java that will execute the regex. The argument "*(#\\w+)\\b*" is the regular expression that represents the hashtag character and characters immediately after that up until a whitespace. The Matcher then aims to look inside the field "tweets[2]" where the tweet is held, and store the word with the hashtag in it. The tweets have been set to lower case because there is a difference between #Rio and #rio for example. Each string of text with a hashtag character in it will be stored and sent off to the Reducer (HashReducer) with an increment of one. The reducer will then aggregate the results for each unique key and write the result.

- Top Hashtag analysis

Rank	Hashtag	No. Occurrences
1	#rio2016	1449246
2	#olympics	91756
3	#gold	68144
4	#bra	50263
5	#futebol	49365
6	#usa	42754
7	#oro	40899
8	#swimming	36649
9	#cerimoniadeabertura	36499
10	#openingceremony	35974

Figure 3 Top 10 Hashtags at the most popular hour

The most used hashtag within the hour of 1am was "#rio2016", which is understandable. One interesting observation is the influence of the Portuguese language on the tweets. #bra or #BRA is the abbreviation for the International Olympic Committee country code for Brazil, and commonly used during sporting events to show support for the nation of Brazil. In addition, *futebol*, in Portuguese, is a direct translation to the English word "football"; As is *cerimonia de abertura* or *cerimônia de abertura,* which is a direct translation to "opening ceremony". It would make sense for the high frequency of Portuguese hashtags/words to be used due to the fact that the Olympic games were held in Brazil, a Portuguese speaking country. The frequent use of the hashtag "#futebol" could also be explained by the celebration of the Brazilian men's football team winning their first gold medal after beating Germany in the final on penalties (Emons, 2016). The most popular sport tweeted at this time was "#futebol", hinting that the football was an activity that took place at this time.

Part C. Support Analysis

1. The aim of Part C.1 is to compute the top 30 athletes based on their support through the Twitter messages.

In the Mapper (AthleteMapper), a Hashtable is introduced, which stores pairs of keys/values.

```
Set<String> AthleteName = athlete.keySet();

for(String name: AthleteName){

if(tweets[2].contains(name)){

data.set(name);
context.write(data,one);
```

Here, the Hashtable is defined by the "athlete" variable. The code sets "AthleteName" to the two fields stored in the Hashtable. The two fields in the Hashtable are the name of the athlete and their respective sport. Looking through all of the tweets, if a name is included, set it to "name", write it and send it off to the Reducer (AthleteReducer) with an increment of one. The Reducer will then, again as it has done for the previous jobs, aggregate like keys and write the keys with their aggregated results.

- Top Athlete Support Analysis

Rank	Athlete Name	Number of Mentions
1	Michael Phelps	178286
2	Usain Bolt	166471
3	Neymar	98094
4	Simone Biles	77093
5	William	50510
6	Ryan Lochte	40363
7	Katie Ledecky	37582
8	Yulimar Rojas	33611
9	Joseph Schooling	25295
10	Sakshi Malik	24565
11	Simone Manuel	23234
12	Rafaela Silva	22576
13	Andy Murray	21086
14	Kevin Durant	20947
15	Tontowi Ahmad	20187
16	Liliyana Natsir	19666
17	Wayde van Niekerk	18194
18	Penny Oleksiak	17266
19	Monica Puig	16003
20	Rafael Nadal	15530

21	Laura Trott	15367
22	Ruth Beitia	13487
23	Teddy Riner	13351
24	Lilly King	13055
25	Shaunae Miller	12094
26	Jason Kenny	11575
27	Allyson Felix	10912
28	Caster Semenya	10799
29	Almaz Ayana	10490
30	Elaine Thompson	10385

Figure 4 Top 30 athletes based on support through tweets

With reference to PartB.2, Michael Phelps (a swimmer) being number 1 on this list, it echoes the support for the sport of swimming as "#swimming" is within the top 10 hashtags in the most popular hour of the day during the Olympic Games. Two Brazilian football players are in the top five athletes in terms of support through tweets: 3. Neymar (mentioned 98,094 times) and 5. William (mentioned 50,510 times), reiterating the claim of overwhelming support due to the men's football team winning their first Olympic gold.

2. The aim of PartC.2 is to compute the top 20 sports according to the mentions of the athletes.

The Mapper (SportMapper) is practically the same as the Athlete Mapper. The only change is as follows:

```
for(String name: AthleteName){
      if(tweets[2].contains(name)){

      data.set(athlete.get(name)); <- CHANGE

      context.write(data,one);
```

From the athlete Hashtable, this will get the sport associated from the athletes. Preceding this, the AthleteCount.java file added a file that portrayed the following information on each medallist:

ID, Name, Nationality, Sex, DOB, Height, Weight, Sport, Gold, Silver, Bronze

This is how the Hashtable is going to work, storing the name of the athletes and the sport from the /data/medalistsrio.csv imported.

- Top Sport Analysis

Rank	Sport	Instances based on Athlete
1	athletics	443018
2	aquatics	436922
3	football	201956
4	gymnastics	124207
5	judo	95058
6	tennis	77380
7	basketball	71675
8	cycling	64557
9	badminton	60372
10	wrestling	33358
11	shooting	22700
12	canoe	22604
13	sailing	22376
14	weightlifting	22095
15	equestrian	21797
16	boxing	20350
17	volleyball	16783
18	rowing	15810
19	taekwondo	15325
20	fencing	11943

Figure 5 Top 20 Sports based on Athlete

The top 3 sports are expected due to the top 3 in the athletes list. Michael Phelps – Aquatics, Usain Bolt – Athletics and Neymar – Football. They aren't the only athletes in those respective sports but they do influence the ranking due to their high mention count. An interesting observation is that athletics takes the top spot here on the Top Sport list but Michael Phelps, an aquatics athlete takes the top spot on the Top Athletes list. It can be fair to assume that across the whole data set, there is more talk of athletics athletes over aquatics athletes.

Throughout the whole assignment, the aim to keep consistency was key to the validity of the results. The condition of the tweet length capped at 140 characters was held consistent throughout the assignment. If that condition was not set at all, the results in Part A and C will definitely be different as they depend on the use of the tweets to perform the MapReduce job.

Important information may be lost due to the if statement capping the characters of the tweets at 140. Foreign languages may use accents and symbols that the file will have to count as a separate character. To reiterate, the logic is held throughout the assignment that the limitation of the tweets is 140 characters, despite Twitter's recent change in its character limit from 140 to 280 (Sulleyman, 2017).

Appendix

1. Results from the MapReduce job in Part B.1 – Time Analysis

Hour	Number of Tweets
0	1335522
1	1677571
2	1592418
3	1274876
4	733462
5	512754
6	463271
7	460629
8	469869
9	479094
10	497403
11	584556
12	780445
13	1041557
14	1244170
15	1325675
16	1304166
17	1359317
18	1408453
19	1434978
20	1522919
21	1451945
22	1288767
23	1324717

References

Emons, M., 2016. *Rio Olympics 2016: Brazil beat Germany on penalties to win men's football gold.* [Online]
Available at: http://www.bbc.co.uk/sport/olympics/36691461
[Accessed 2017].

Lyons, S., 2016. *A Day-by-Day Schedule of the 2016 Rio Olympics.* [Online]
Available at: http://www.vulture.com/2016/08/olympics-2016-schedule-event-guide.html

Sulleyman, A., 2017. *Twitter introduces 280 characters to all users.* [Online]
Available at: http://www.independent.co.uk/life-style/gadgets-and-tech/news/twitter-280-characters-tweets-start-when-get-latest-a8042716.html

Twitter, 2017. *Posting links in a Tweet.* [Online]
Available at: https://support.twitter.com/articles/78124