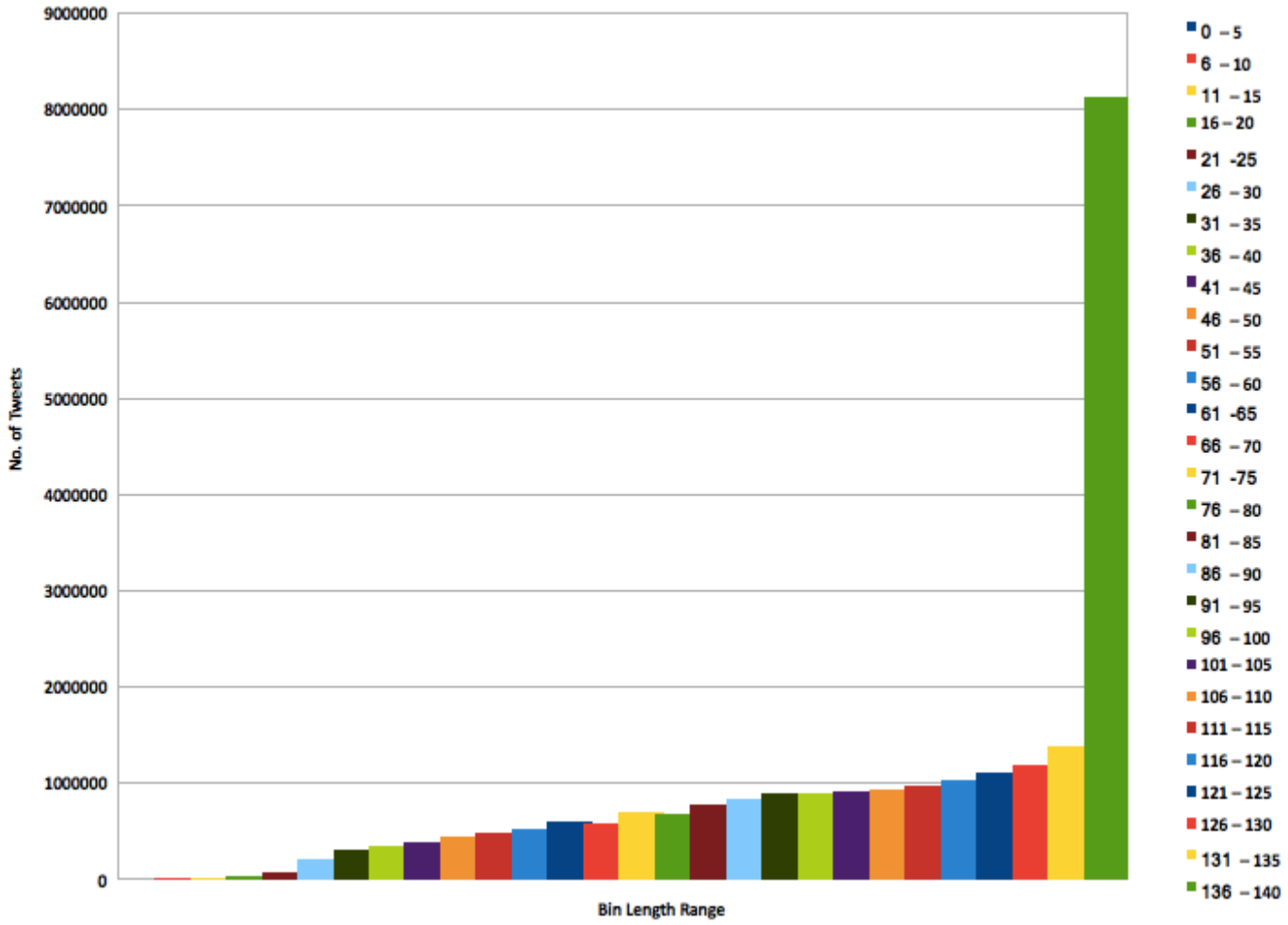
Twitter Analysis using MapReduce
Big Data Processing – Assignment 1

The main aim of this report is to provide a comprehensive big-data analysis of the millions of tweets during 2016 Rio Olympics. The supporting evidence of code and source files have been attached in a separate folder along with this report.

The report is divided into three parts – A) Message Length Analysis, Time Analysis and Support Analysis. It is also expected that the all the tweets which are not in the following format be filtered: Epoch_time; tweet_ID; tweet (including hashtags) ; device_used. Thus, the analysis has only been done to the filtered tweets. The filtration condition has been given in the mapper for the various jobs.

Part A – Message Length Analysis

In this section, the length of the message is aggregated using MapReduce in Hadoop. Firstly, the twitter input is parsed as part of the filtering using the String.length() method which is separated by a ";" and imported the appropriate java (StringUtils) package. The relevant field is then extracted from the String, in this case field [2] for the tweet message. It is further given that; the maximum length of the message cannot be greater than 140 characters as per the guidelines. Thus, a condition is given in the TweetLengthMapper, where different bins are aggregated for different length ranges. It starts from 1-5, 6-10 until 136-140 i.e. with a range of +5 starting from the 1st character. The bins have been aggregated using the '%' – modulus operator. Firstly, I've assigned a nested if else condition where if the characters <= 140, the bin length would be set to length/5 -1, if length %5 =0. Or else, the bin length would be set to length/5. Since, in java numbering starts from 0, we have to subtract 1 from the length, so that it falls in the first bin range of 1-5, synonymous to bin 0. Example, if the length of the tweet is 5, it would fall in bin 1-5, which would be bin 0. Suppose the length is 6, then it would fall in bin 1 (6-10). Since, when %>0, it is rounded down to the lowerbound integer. Once, the mapper collects the keys (bins) and the values (1)(the number that fall in each bin), they are sent to the TweetLengthReducer, to aggregate the total values for the different keys.



I've plotted a histogram using the output from the Hadoop job. The keys (bins 0-27) have been renamed using the ranges 1-5 to 136 – 140 in excel. The MapReduce output has been included in the job folder. As can be seen in the histogram, the number of tweets that fall in the various bin length ranges have been captured. It is obvious from the graph, that majority of the tweets (around 8.1 million) lie in the 136-140 characters range. This also implies, that many of the tweets that are greater than 140 characters have been omitted. One can draw insights from this that many tweets having a greater length than 140 characters have been left out due to the filtering process.

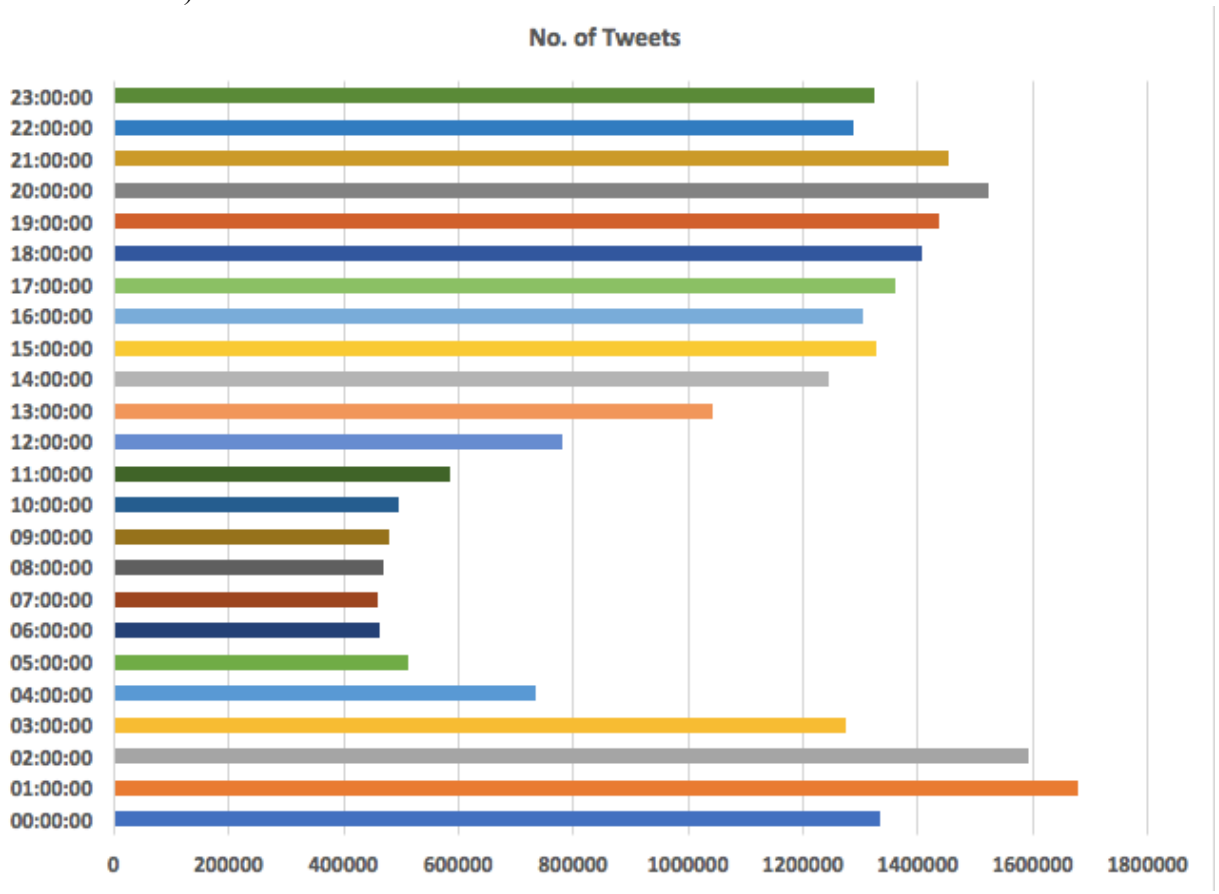
Part B – Time Analysis

For part B, the basic filtering where the tweets are in the specified format is done. However, I've computed the tweets per hour and the hashtags without formatting the message length to 140 characters.

1) Number of Tweets every hour

For Part B1, the number of tweets for every hour of the day (24 hour) regardless of the day has been computed. Most part of this job has been done in the TweetTimeMapper. Like in part A, the input has been parsed using StringUtils to filter the tweet then the relevant field – the epoch time – field [0] has been extracted from string. From the epoch time, the tweets occurring at that specific hour would be parsed. Using LocalDateTime and ZoneOffset, SimpleDateFormat, the tweets for that specific hour have been passed. The EpochTime which has been given in milliseconds has been converted to seconds using LocalDateTime.ofEpochSecond(EpochTime/1000). Furthermore, the local time in Rio has been converted to UTC time using ZoneOffset. A try-catch statement has been used to exclude exceptions that did not have an EpochTime. Without listing for the exception, the MapReduce job would fail even if the build was successful. Example: foreign messages where the tweet message starts with the # - "#Argentina le ganó a #Rusia, ultimo campeón olímpico 25-18". The key-value pairs from the mapper

have been emitted in the form of time:hours and ones(1). These values have been summed up for the different keys in the TweetTimeReducer. The results were then depicted on a bar plot in Excel. It can be seen that the most number of tweets (were posted during 1:00 am UTC time with more than 1.6 million tweets).



2) Top 10 Hashtags at the most popular Hour (1:00 am)

For the second part, I've repeated the same procedure as the first to find the tweets for every hour first. However, I've hardcoded the most popular hour into the PeakTimeMapper and then used pattern matching for the hashtags (start with # substrings). I've used the regex pattern of \w = [a-zA-Z_0-9] and ending within the word boundary \b to identify the hashtags. Subsequently, the emitted [key, value] pairs from the mapper have been summed in the PeakTimeReducer to give the final output keys and values. The top ten hashtags have been sorted from the results using the linux sort | head command.

	Hashtags	No. of Tweets
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

Part C – Support Analysis

For Part C, the main tweets dataset has to be joined with the secondary medalists dataset that has been given. For this job, a replication join method has been used where most of the work is done in the Mapper and the secondary dataset to be joined has to be added to the path in the main Job file. A Hashtable has been used and the KeySet method has been implemented to set the key in the tweet dataset. Here as well, the main dataset has been filtered where the tweets are in the specified format. However, the message length of the tweet has not been formatted to 140 characters.

1) Top 30 Athletes

To compute the top 30 athletes based on their mentions in tweets, the medalists dataset was joined with the main tweets file in the job file and the methods for the join and hashtable have been set in the TwitterAthletesMapper. It is also given, that the names in the tweet should be exactly as given in the dataset with the first name and last name. After the general filtration process using StringUtils like in the previous parts, the relevant field, the message field [2] has been parsed. Then, the keyset method has been used to identify the athlete names in the tweet. For the Hashtable, the fields [0] and [7] have been selected as the key, value pairs from the secondary dataset that correspond to athlete name and sport respectively. Since, we are concerned with only the names in the first part, we join the datasets based on the key. The emitted key, value pairs from the mapper would be the (athlete name, 1) which would then be aggregated in the TwitterAthletesReducer. A separate map-side join and using that output as input for a map-reduce job could be another way of processing the data. However, since the data is not in sequencefileformat and the output format remains the same, both the join and reduce could be done in one job. Finally, the top 30 athletes have been computed from the output after sorting it in Excel.

	Athlete	**Tweet-Mention**
1	Michael Phelps	181167
2	Usain Bolt	170647
3	Neymar	100853
4	Simone Biles	79300
5	William	53262
6	Ryan Lochte	40773
7	Katie Ledecky	37885
8	Yulimar Rojas	34443
9	Simone Manuel	27367
10	Joseph Schooling	26467
11	Sakshi Malik	24644
12	Rafaela Silva	22805
13	Andy Murray	21776
14	Kevin Durant	21263
15	Tontowi Ahmad	20428
16	Liliyana Natsir	19905
17	Wayde van Niekerk	18343
18	Penny Oleksiak	17575
19	Monica Puig	17208
20	Rafael Nadal	16120
21	Laura Trott	16098
22	Ruth Beitia	14930

23	Teddy Riner	13995
24	Lilly King	13279
25	Shaunae Miller	12250
26	Jason Kenny	12140
27	Elaine Thompson	12111
28	Caster Semenya	11675
29	Almaz Ayana	11092
30	Allyson Felix	11066

2) Top 20 Sports

For the second part, the top 20 sports have to be computed based on the names of athletes mentioned in the tweets. The same procedure as for the preceding question has been followed. Except, in the keySet instead of setting the key [sport.set(athleteName)] we select the value [sport.set(athlete.get(athleteName))]. The key value pairs [sport, one] are then aggregated in the reducer to give the final results.

	Sport	**Athlete-Tweet**
1	athletics	458327
2	aquatics	449605
3	football	208687
4	gymnastics	127780
5	judo	97214
6	tennis	81300
7	basketball	73262
8	cycling	66787
9	badminton	61006
10	wrestling	33979
11	canoe	23952
12	shooting	23928
13	sailing	23921
14	weightlifting	23766
15	equestrian	23197
16	boxing	23159
17	rowing	17394
18	volleyball	17208
19	taekwondo	15956
20	fencing	12716

Conclusion

The Message Length Analysis, Time Analysis and the Support Analysis of the tweets during Rio Olympics 2016 has been conducted.

References

- Cuadrado, F. (2017). *Big Data Processing Lab Material.*
- Cuadrado, F. (2017). *Big Data Processing Lectures*.
- Docs.oracle.com. (2017). *Date (Java Platform SE 7)*. [online] Available at: https://docs.oracle.com/javase/7/docs/api/java/util/Date.html [Accessed 10 Nov. 2017].
- Docs.oracle.com. (2017). *Hashtable (Java Platform SE 7)*. [online] Available at: https://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html [Accessed 10 Nov. 2017].
- Docs.oracle.com. (2017). *Pattern (Java Platform SE 7)*. [online] Available at: https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html [Accessed 10 Nov. 2017].
- StackOverFlow.com (2015). *Check if String contains hashtag word*. [online] Stackoverflow.com. Available at: https://stackoverflow.com/questions/34047165/check-if-string-contains-hashtag-word [Accessed 10 Nov. 2017].
- StackOverFlow.com (2015). *Get LocalDateTime from seconds including the timezone*. [online] Stackoverflow.com. Available at: https://stackoverflow.com/questions/32317984/get-localdatetime-from-seconds-including-the-timezone [Accessed 10 Nov. 2017].