BIG DATA PROCESSING

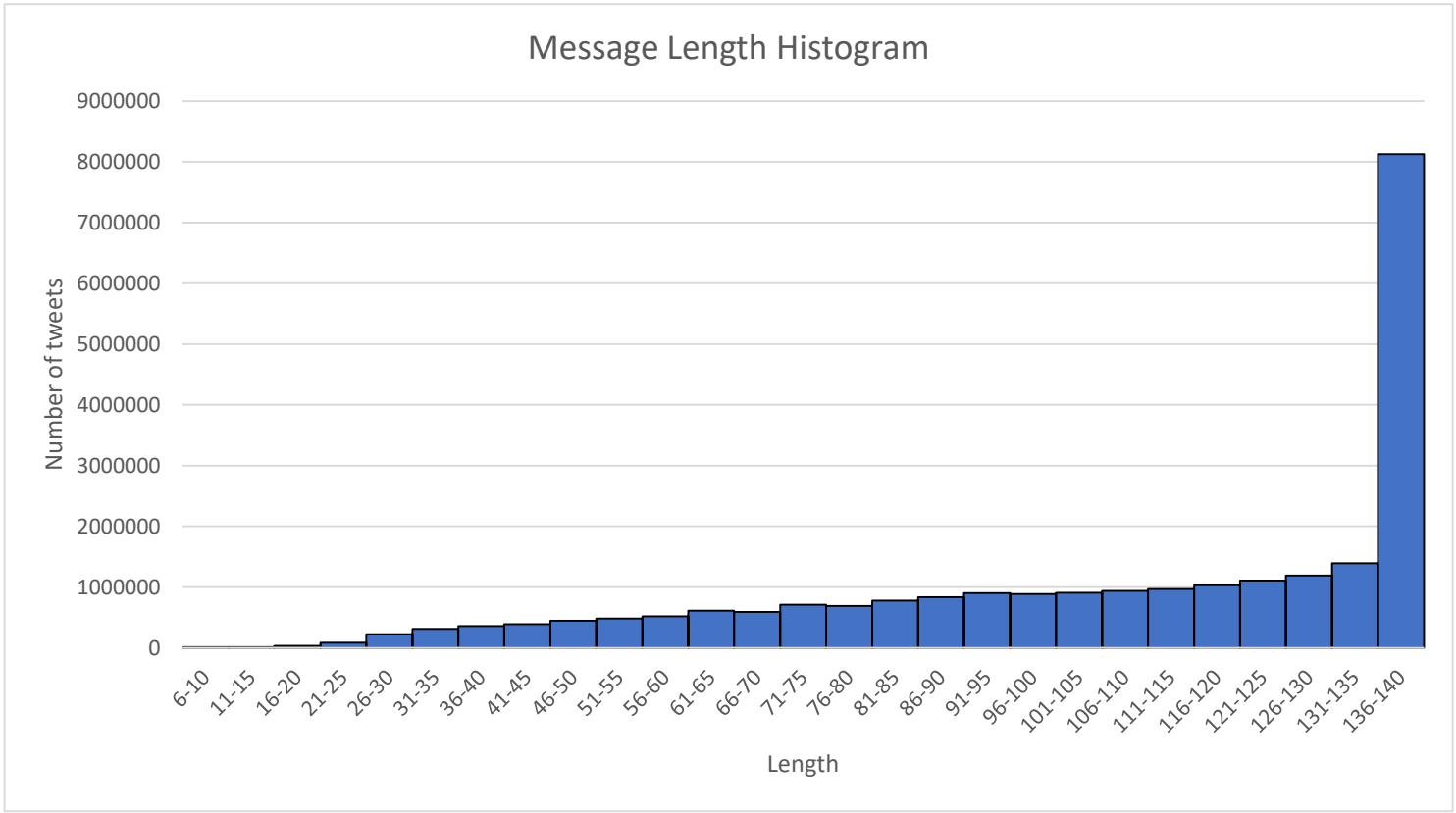
Bayzid Chowdhury

150245579

Part A:

I created three java files to compute the tweet message length. The map method in the LengthMapper class will split the dataset text line using semi colon and equate it to a string array. Then it will check if the length of the array is four and length of the message is less than 141, if it's true, it will call the method getRange() where this method takes in the length of the tweet message by calling length() on the string object. In the method getRange() there are 2 variables to keep record of the group range, the while loop will execute until the minimum value is less than 141. The if statement will check if the length is between the minimum and maximum, if it's true, it will break from the while loop and return the range, else it will keep adding 5 to the minimum and maximum until a range is found. The mapper will emit the key as Text range and value as IntWritable one. The reducer will loop through IntWritable values and tally up the sum to generate the key and the total sum.

Histogram plot: The results below shows that the largest number of tweets is in the 136-140 message length range. Also for the first 3 ranges, the number of tweets are too small to display on the graph.

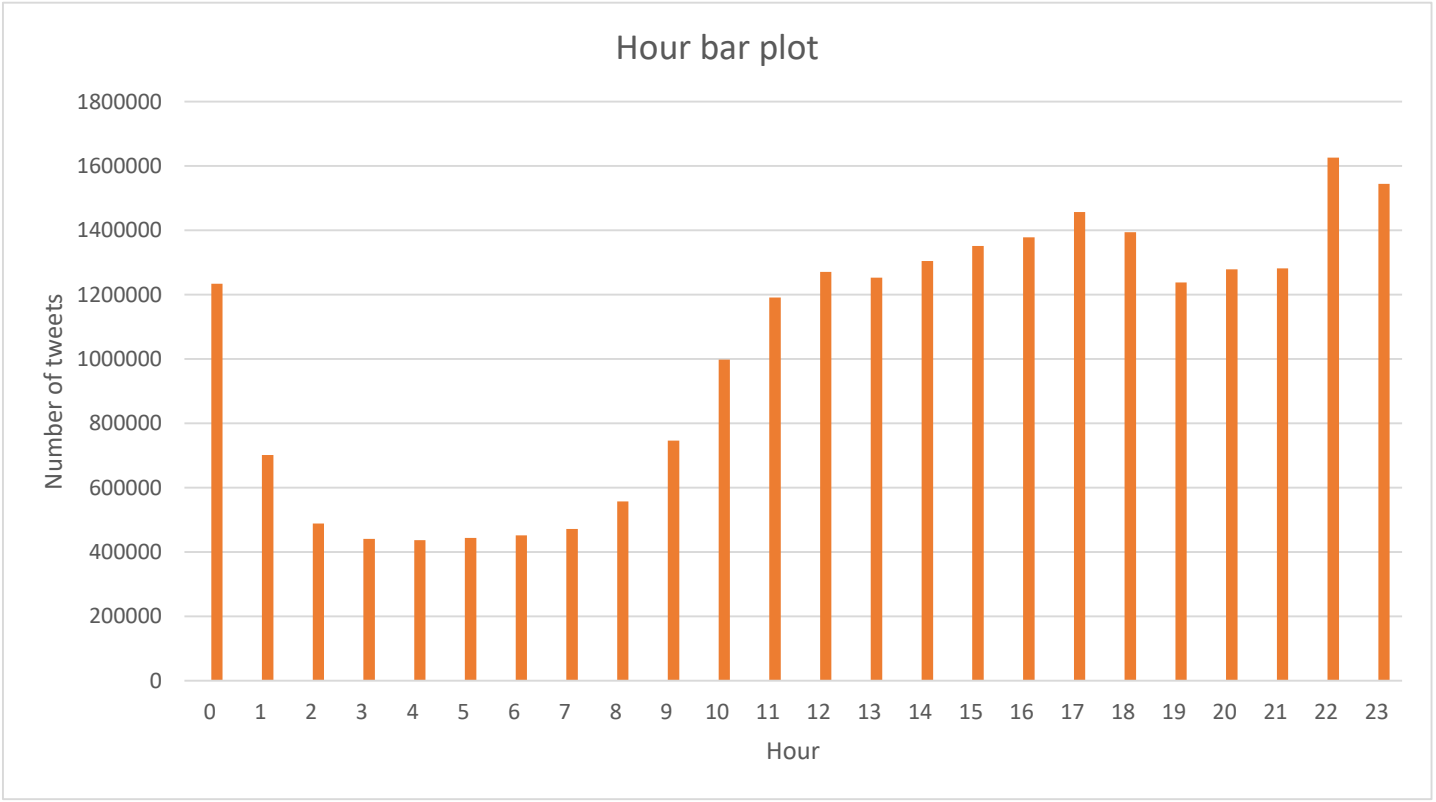


Part B:

Part 1:

I created three java files to compute the number of tweets that were posted each hour of the event. The method map in the class TimeMapper will first split the dataset text line using semi colon and use the first index to retrieve the time. The method getEachHour() takes in the time as type long. The ofEpochSecond() method takes 3 arguments and returns a LocalDateTime object, then the hour can get retrieved by calling getHour() method on the LocalDateTime object. I put try catch block where epoch time is being parsed so the job wouldn't fail due to error in the first index of the input line. The 3rd argument in the ofEpochSecond() method took in zoneOffset.ofHours(-3) as there was a 3 hour difference in timezone between here and Brazil when Olympics occurred. The mapper will emit the key as IntWritable hour and the value IntWritable one. The reducer loops through the values and will tally up the total sum like previous reduce code.

Bar plot: Below is the result showing that the largest number of tweets occurred in the 22nd hour.



Part B:

Part 2:

I created three java files to compute the top 10 hashtags for the most popular hour. Similar to part 1 we get the hour of the message. From part 1 we found that 22nd hour was the most popular hour so we only look for hashtags in that specific hour. I used regular expression to only include messages which start with hashtag(#) followed by one or more English letters or numbers. This was used to filter out other languages and emoji's from the tweet. In the while loop, the method find() is called on the Matcher object, this will return true as long as there is a next subsequence that match the regular expression. Then group() will be called on the Matcher object to return the string that was matched, which the mapper emits as key as Text and the value as IntWritable one. The reducer loops through the values and will tally up the total sum like previous reduce code.

Table: results are shown below, this was sorted using excel. From this result we can see that opening ceremony might have taken place at that hour, also there was a lot of support for USA by looking at the #USA and #TeamUSA hashtags.

Top 10 Hashtags	
#Rio2016	1320441
#Olympics	76598
#rio2016	71493
#Futebol	40195
#USA	39336
#Gold	37323
#BRA	36630
#CerimoniaDeAbertura	33610
#OpeningCeremony	33493
#TeamUSA	30216

Part C:

Part 1:

I created three java files to compute the top 30 athletes. I added a cache in the class Athletes which gets the medalistsrio.csv file, which now allows me to join 2 input files. I override the setup method in the AthletesMapper class which reads from the csv file and it splits each line by comma. If the length of the array is 11 and the message is not empty, it will add the athlete's name to the arraylist. Now all the athlete's name will be stored in the arraylist which the map method will use to find names in tweets. Mapper will loop through all the names in arraylist and try find if any names from the arraylist are mentioned in the tweet. This is done by using the method contains(). I also checked for tweets that may have names of athletes in mixture of upper or lower cases. So if a tweet had "NEYMAR" all upper case, by using the toLowerCase() method I was able to do the comparison while both strings are in lower case, so any combination of cases will still be a match. The mapper emits the athlete's name as key of Text and value IntWritable one. The reducer loops through the values and will tally up the total sum like previous reduce code.

Table: results are shown below, this was sorted using excel. From this result we can see that Michael Phelps was mentioned the most.

Top 30 athletes	
Michael Phelps	187921
Neymar	173241
Usain Bolt	171800
Simone Biles	79777
William	61637
Ryan Lochte	41276
Katie Ledecky	39576
Yulimar Rojas	34748
Rafaela Silva	25830
Joseph Schooling	25816
Sakshi Malik	25122
Simone Manuel	24268
Andy Murray	21617
Wayde van Niekerk	21561

Kevin Durant	21193
Tontowi Ahmad	20911
Liliyana Natsir	20257
Andre de Grasse	17921
Penny Oleksiak	17797
Monica Puig	17484
Rafael Nadal	16144
Laura Trott	15632
Ruth Beitia	14475
Lilly King	14185
Teddy Riner	14079
Luan	13796
Shaunae Miller	12243
Jason Kenny	11995
Caster Semenya	11153
Allyson Felix	11114

Part C:

Part 2:

I created three java files to compute the top 20 sports according to the athletes mentioned. I added a cache in the class Sports same as part 1. Everything is the same as part 1 but with sport's name, I added another arraylist which holds the sport's name that is read from the medalistsrio.csv file. It will do the same computation as part 1 i.e. it will loop through all the names in arraylist and try find if any names from the arraylist are mentioned in the tweet. If a name is found then the mapper will not emit the athlete's name but will emit the sport's name as key of Text and the value IntWritable one. The reducer loops through the values and will tally up the total sum like previous reduce code.

Table: results are shown below, this was sorted using excel. From this result we can see that athletics was most popular.

Top 20 sports according to athletes mentioned	
athletics	477845
aquatics	458416
football	300298
gymnastics	134322
judo	102935
tennis	84707
basketball	74767
cycling	69064
badminton	63158
wrestling	35143
weightlifting	23836
sailing	23405
canoe	23219
shooting	23101
equestrian	22628
boxing	20964
volleyball	17614
rowing	16348
taekwondo	15694
fencing	12888