

ECS765 Big Data Processing

Coursework 1: Twitter Analysis with Map Reduce

***Student Name, Student ID:***

*Adebisi Afolalu, 120693551*

***Degree Course, Module:***

*MSc Big Data Science, ECS765*

Table of Contents

[1. Part A: Message Length Analysis 2](#_Toc498121752)

[1.1. MapReduce 2](#_Toc498121753)

[1.1.1. Mapper 2](#_Toc498121754)

[1.1.2. Reducer 3](#_Toc498121755)

[1.2. Results: Frequency of Message Length 4](#_Toc498121756)

[1.3. Histogram Plot 5](#_Toc498121757)

[2. Part B: Time Analysis 6](#_Toc498121758)

[2.1. MapReduce 6](#_Toc498121759)

[2.1.1. Mapper 6](#_Toc498121760)

[2.1.2. Reducer 7](#_Toc498121761)

[2.2. Results: Frequency of Tweets each Hour 8](#_Toc498121762)

[2.3. Bar plot 9](#_Toc498121763)

[2.4. Results: Top Ten Hash Tags 10](#_Toc498121764)

[3. Part C: Support Analysis 11](#_Toc498121765)

[3.1. MapReduce 11](#_Toc498121766)

[3.1.1. Mapper 11](#_Toc498121767)

[3.1.2. Reducer 12](#_Toc498121768)

[3.2. Results: Top 30 Athletes Mentioned 13](#_Toc498121769)

[3.3. Results: Top 20 Sports Mentioned 14](#_Toc498121770)

Table of Figures

[Figure 1: Snippet from Message Length Mapper code 2](#_Toc498121771)

[Figure 2: Snippet from Message Length Reducer code 3](#_Toc498121772)

[Figure 3: Histogram for distribution of tweet message lengths 5](#_Toc498121773)

[Figure 4: Snippet from Time Analysis Mapper code 6](#_Toc498121774)

[Figure 5: Snippet from Time Analysis Reducer code 7](#_Toc498121775)

[Figure 6: Snippet from Support Analysis Mapper code 11](#_Toc498121776)

[Figure 7: Extracting required columns in medalistrio.csv 11](#_Toc498121777)

[Figure 8: Snippet from Support Analysis Reducer code 12](#_Toc498121778)

# Part A: Message Length Analysis

## MapReduce

### Mapper



Figure 1: Snippet from Message Length Mapper code

The Mapper job processes the Olympic tweets (input data) that’s stored on the Hadoop Distributed File System (HDFS). It takes the input data line by line, computing the correct bin for each tweet.

As shown in the first line of code in fig. 1, the split method was used to divide the input data into the tweet elements (epoch\_time, tweet ID, tweet message and device type) on the delimiter “;”. The divided elements were stored in an array. An if statement was used to iterate over the divided elements and retrieve the four elements that form the tweet.

The “tweetLen” variable in fig. 1 stored the extracted 3rd element of every 4 with length method applied. The 3rd element (index 2 of the array) was the tweet message and the length method computed the total number of characters in the message. An if statement nested within the first if statement filtered out any tweet message without a valid length (of 1, 140 or any value in between. When the if condition was met the correct bin was calculated.

The bins for each valid tweet was calculated by dividing the message length by the class width 5. This simplified the problem and assigned each tweet with a bin between 1 and 28. The ceil function was applied to round up the tweet length/5 float point to the nearest integer value. This ensured that each tweet was assigned to the correct bin (e.g. message length 136 was assigned to bin 28).

The processed data is emitted through context.write() to the reducer, as smaller chunks of key/value pairs in a serialised fashion. The first argument K2 and second argument V2 of the context.write() function represented the bin and one respectively.

### Reducer

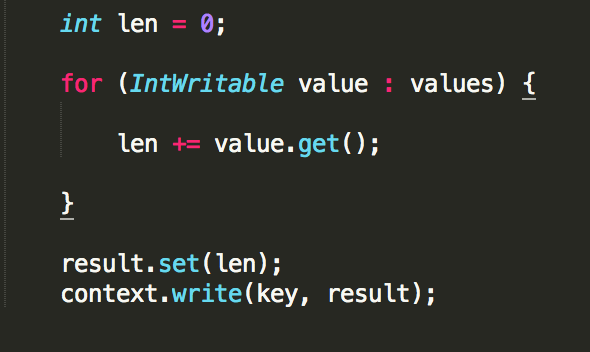


Figure 2: Snippet from Message Length Reducer code

After the shuffle and sort phase, the Reducer job receives K2/V2 pairs from the mapper as its inputs. A for loop is executed for each K2/V2 pair. The for loop simply sums the frequency of each bin with a counter that increments by 1 for each occurrence.

The context.write function emits K3 and V3 as a list, where K3 is the bins and V3 is the frequency of the corresponding bin. The list of K3/V3 pairs is saved on the Hadoop Distributed File System (HDFS).

## Results: Frequency of Message Length

The MapReduce job aggregated the bins, which was saved in a text file. Excel was used to import the data in the text file and sort by bins in ascending order. Labels was then assigned to each corresponding bin e.g. bin 28 was labelled “136 – 140”.

|  |  |  |
| --- | --- | --- |
| **Bins** | **Class (Tweet Lengths)** | **Frequency** |
| 1 | 1 - 5 | 11200 |
| 2 | 6 - 10 | 7645 |
| 3 | 11 - 15 | 26024 |
| 4 | 16 - 20 | 73461 |
| 5 | 21 - 25 | 181833 |
| 6 | 26 - 30 | 309019 |
| 7 | 31 - 35 | 345659 |
| 8 | 36 - 40 | 386889 |
| 9 | 41 - 45 | 428841 |
| 10 | 46 - 50 | 481598 |
| 11 | 51 - 55 | 497100 |
| 12 | 56 - 60 | 587357 |
| 13 | 61 - 65 | 610618 |
| 14 | 66 - 70 | 709109 |
| 15 | 71 - 75 | 683743 |
| 16 | 76 - 80 | 746501 |
| 17 | 81 - 85 | 806405 |
| 18 | 86 - 90 | 875538 |
| 19 | 91 - 95 | 918900 |
| 20 | 96 - 100 | 923181 |
| 21 | 101 - 105 | 917333 |
| 22 | 106 - 110 | 959870 |
| 23 | 111 - 115 | 1023777 |
| 24 | 116 - 120 | 1076193 |
| 25 | 121 - 125 | 1181781 |
| 26 | 126 - 130 | 1337609 |
| 27 | 131 - 135 | 2021925 |
| 28 | 136 - 140 | 6411788 |

## Histogram Plot

Excel was used to generate the histogram for the aggregated bins. It represents the distribution of the tweet message lengths, ready to support deeper data analysis.

Figure 3: Histogram for distribution of tweet message lengths

# Part B: Time Analysis

## MapReduce

### Mapper

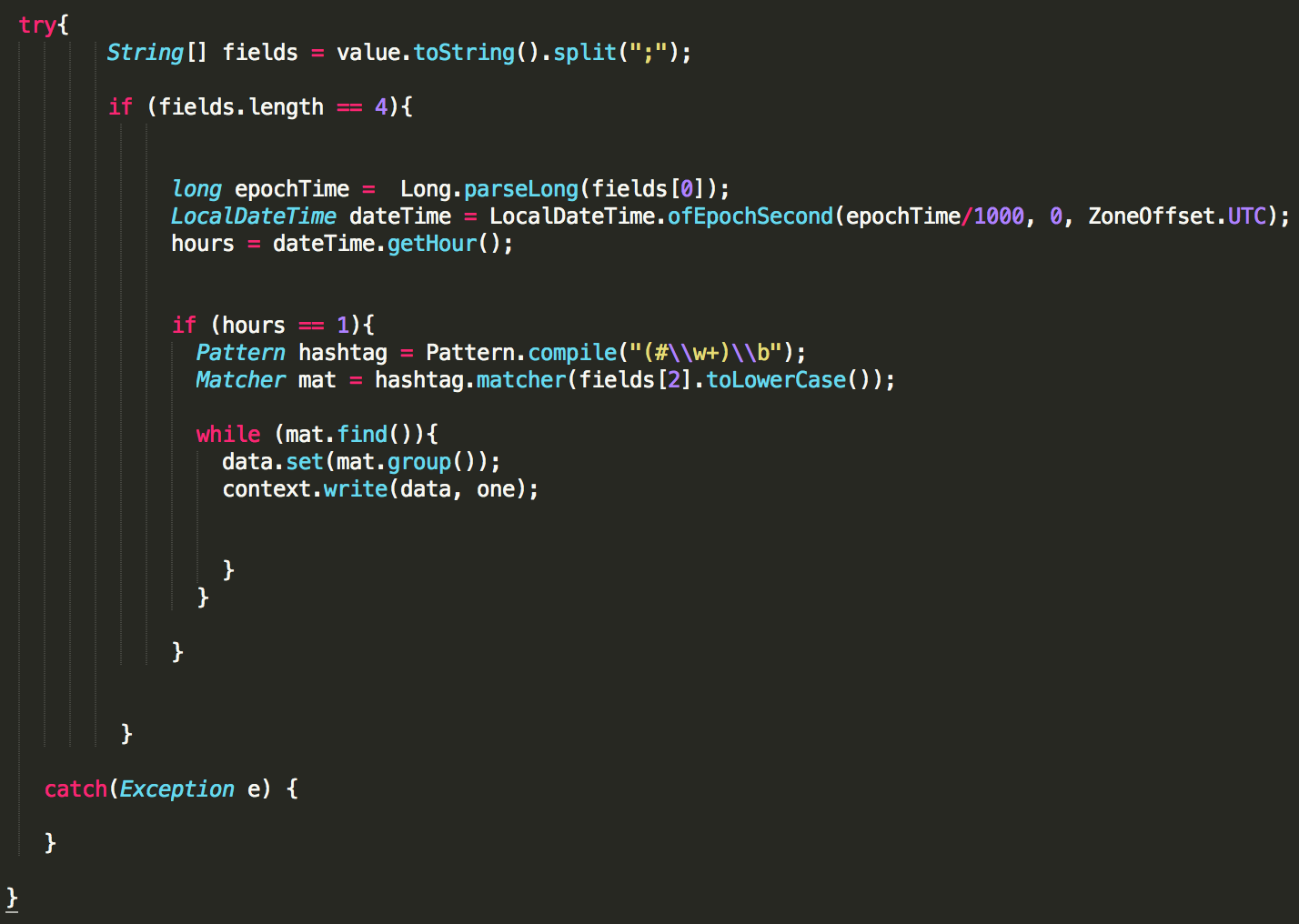


Figure 4: Snippet from Time Analysis Mapper code

The Mapper job processes the Olympic tweets (input data) that’s stored on the Hadoop Distributed File System (HDFS). It takes the input data line by line, computing number of tweets that were posted each hour of the event. It then filtered the tweet messages that occurred at 1am (the most popular hour of the Olympic games) leaving only the hashtags used.

As shown as the first try statement in fig. 4, the split method was used to divide the input data into the tweet elements (epoch\_time, tweet ID, tweet message and device type) on the delimiter “;”. The divided elements were stored in an array. An if statement was used to iterate over the divided elements and retrieve the four elements that form the tweet.

The “epochTime” variable in fig. 4 stored the extracted 1st element of every 4. The 1st element (index 0 of the array) contained the epoch time expressed in milliseconds. The epoch time in each tweet is declared as a float point data type and then divided by 1000 to convert the units to seconds.

The “ofEpochSecond” function is used to compute the data and time since 01/01/1970. This was used to convert the epoch time of each tweet into a date and time, in the format YYYY-MM-DD HH:MM:SS. The time zone was set to UTC for outputting the local time in the UK. The “getHour” method was used to get the specific hour for each tweet, based on the output of the “ofEpochSecond”.

An if statement nested within the first if statement filtered out tweets, excepts the tweets that occurred at 1am (the most popular hour of the games). The Pattern.compile function is used to define any hashtag (pattern), characterised by a hash symbol and the subsequent characters before a space. The matcher method matches the defined hashtag to the tweet message (index 2 of the tweet array). The find and group methods are used to search for occurrences of the defined pattern and repeats the process for each tweet message in the input data.

The processed data is emitted through context.write() to the reducer, as smaller chunks of key/value pairs in a serialised fashion. The first argument K2 and second argument V2 of the context.write() function represented the hashtags found and one respectively.

### Reducer

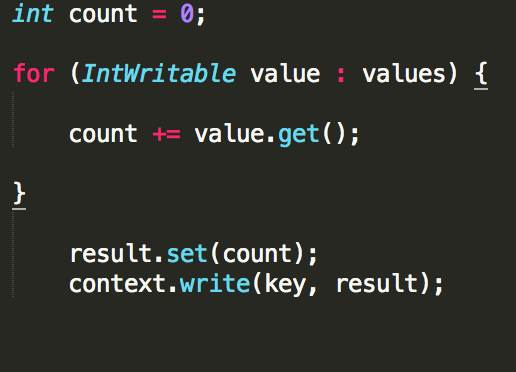


Figure 5: Snippet from Time Analysis Reducer code

After the shuffle and sort phase, the Reducer job receives K2/V2 pairs from the mapper as its inputs. A for loop is executed for each K2/V2 pair. The for loop simply sums the frequency of the tweets or each hashtag with a counter that increments by 1 for each occurrence.

The context.write function emits K3 and V3 as a list, where K3 is the hashtags and V3 is the frequency of the corresponding hashtags. The list of K3/V3 pairs is saved on the Hadoop Distributed File System (HDFS).

## Results: Frequency of Tweets each Hour

The MapReduce job aggregated together all the messages sent at the same hour, which was saved in a text file. Excel was used to import the data in the text file and sort by hours in ascending order.

|  |  |
| --- | --- |
| **Hours** | **Total 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 | 1244185 |
| 15 | 1325675 |
| 16 | 1304166 |
| 17 | 1359317 |
| 18 | 1408453 |
| 19 | 1434978 |
| 20 | 1522919 |
| 21 | 1451945 |
| 22 | 1288767 |
| 23 | 1324717 |

## Bar plot

Excel was used to generate the bar plot for the frequency of tweets each hour. This allows us to make quick and accurate conclusions about the data, including that the prime time of tweets about the events was between 1am and 2am (UK time).

## Results: Top Ten Hash Tags

The MapReduce job aggregated together all the hashtags used within tweet messages between 1am and 2am, which was saved in a text file. Excel was used to import the data in the text file and sort by frequency in descending order. The table below shows the top ten hashtags of the hour.

|  |  |
| --- | --- |
| **Hash Tag** | **Frequency** |
| #rio2016 | 1449246 |
| #olympics | 91756 |
| #gold | 68144 |
| #bra | 50263 |
| #futebol | 49365 |
| #usa | 42754 |
| #oro | 40899 |
| #swimming | 36649 |
| #cerimoniadeabertura | 36499 |
| #openingceremony | 35974 |

# Part C: Support Analysis

## MapReduce

### Mapper



Figure 6: Snippet from Support Analysis Mapper code

The Mapper job processes the Olympic tweets (input data) that’s stored on the Hadoop Distributed File System (HDFS). It takes the input data line by line, computing number of mentions for athlete names or sports in the tweet message.

The split method was used to divide the input data into the tweet elements (epoch\_time, tweet ID, tweet message and device type) on the delimiter “;”. The divided elements was stored in an array. An if statement was used to iterate over the divided elements and retrieve the four elements that form the tweet. A nested for loop was used iterate over the rows in the second dataset medalistrio.csv. The required columns name (index 1) and sport (index 7) was selected with the put method, as shown in fig. 7. The if statement nested within the for loop was used emit all athlete names or sports mentioned in the tweet message.

The processed data is emitted through context.write() to the reducer, as smaller chunks of key/value pairs in a serialised fashion. The first argument K2 and second argument V2 of the context.write() function represented the athlete names or sports mentioned and one respectively.

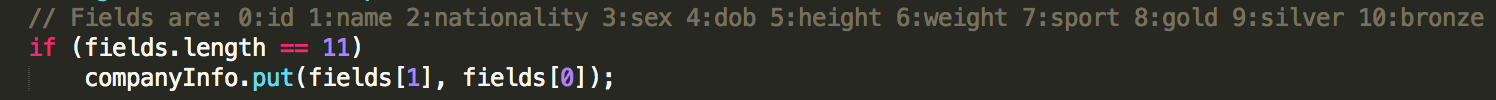


Figure 7: Extracting required columns in medalistrio.csv

### Reducer

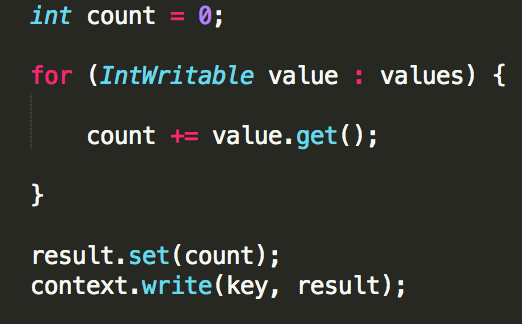


Figure 8: Snippet from Support Analysis Reducer code

After the shuffle and sort phase, the Reducer job receives K2/V2 pairs from the mapper as its inputs. A for loop is executed for each K2/V2 pair. The for loop simply sums the frequency of mentions for athlete names or sport with a counter that increments by 1 for each occurrence.

The context.write function emits K3 and V3 as a list, where K3 is the hashtags and V3 is the frequency of the corresponding athlete names or sport. The list of K3/V3 pairs is saved on the Hadoop Distributed File System (HDFS).

## Results: Top 30 Athletes Mentioned

The MapReduce job aggregated together all the athletes mentioned within tweet messages, which was saved in a text file. Excel was used to import the data in the text file and sort by total mentions in descending order. The table below shows the top 30 athletes based on mentions.

|  |  |
| --- | --- |
| **Athletes** | **Total Mentions** |
| Michael Phelps | 181167 |
| Usain Bolt | 170647 |
| Neymar | 100853 |
| Simone Biles | 79300 |
| William | 53262 |
| Ryan Lochte | 40773 |
| Katie Ledecky | 37885 |
| Yulimar Rojas | 34443 |
| Simone Manuel | 27367 |
| Joseph Schooling | 26467 |
| Sakshi Malik | 24644 |
| Rafaela Silva | 22805 |
| Andy Murray | 21776 |
| Kevin Durant | 21263 |
| Tontowi Ahmad | 20428 |
| Liliyana Natsir | 19905 |
| Wayde van Niekerk | 18343 |
| Penny Oleksiak | 17575 |
| Monica Puig | 17208 |
| Rafael Nadal | 16120 |
| Laura Trott | 16098 |
| Ruth Beitia | 14930 |
| Teddy Riner | 13995 |
| Lilly King | 13279 |
| Shaunae Miller | 12250 |
| Jason Kenny | 12140 |
| Elaine Thompson | 12111 |
| Caster Semenya | 11675 |
| Almaz Ayana | 11092 |
| Allyson Felix | 11066 |

## Results: Top 20 Sports Mentioned

The MapReduce job aggregated together all the sports mentioned within tweet messages, which was saved in a text file. Excel was used to import the data in the text file and sort by total mentions in descending order. The table below shows the top 20 sports based on mentions.

|  |  |
| --- | --- |
| **Sports** | **Total Mentions** |
| badminton | 325781 |
| volleyball | 280069 |
| basketball | 188228 |
| judo | 141142 |
| gymnastics | 140805 |
| tennis | 135169 |
| football | 130891 |
| hockey | 107961 |
| handball | 93221 |
| cycling | 79909 |
| rowing | 59023 |
| wrestling | 57560 |
| golf | 55657 |
| athletics | 53799 |
| boxing | 49316 |
| fencing | 39338 |
| shooting | 38820 |
| weightlifting | 36684 |
| triathlon | 28888 |
| sailing | 21579 |