Inês de Almeida Nolasco

A-Twitter message length analysis:

Goal: Create a histogram of the tweets message's length where each bin aggregates the lengths in groups of 5.

- ## Posing the problem as a Map-reduce problem:

The desired output of this map-reduce program is a list of bins each followed by the count of how many messages have a length that falls inside that bin. The bins are required to join together groups of 5 different lengths, so we have the particularity that a unique bin does not represent a unique length.

Without this requirement, this problem can be seen exactly as the WordCount problem discussed in the lectures. In WordCount, each unique word is used as a key and the occurrence of a word in the text creates a pair (word, 1), the pairs are then sent to the reducers, where all the occurrences of the same unique Key(word) end up together and are summed up to compute the total number of occurrences of that word.

Here, as in WordCount, we need to count the occurrence of some feature, which in this case is message length. The idea would then be that for each message arriving to the mapper, the length is computed and the pair (length, 1) is emitted to the reducers. The Reducers would take care of summing all these 1 values to compute the number of occurrences of each length.

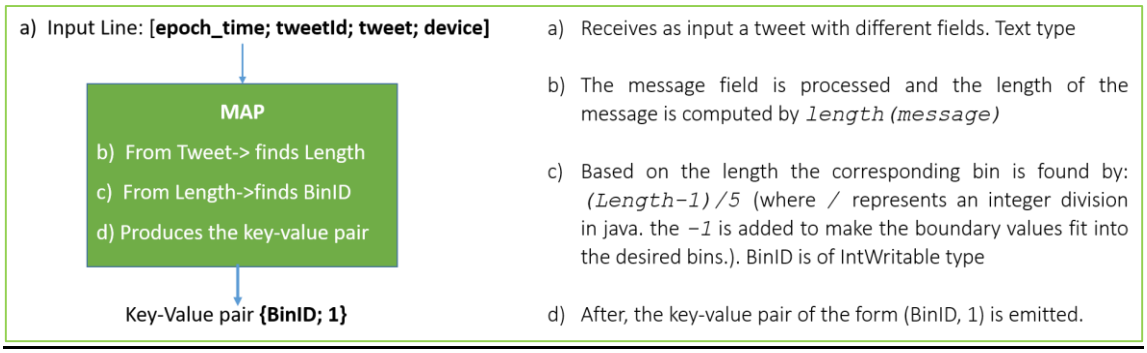
Adding the requirement mentioned above, implies that the final aggregation should be done, not around each unique message length, but instead around a unique bin of the final histogram we wish to output.

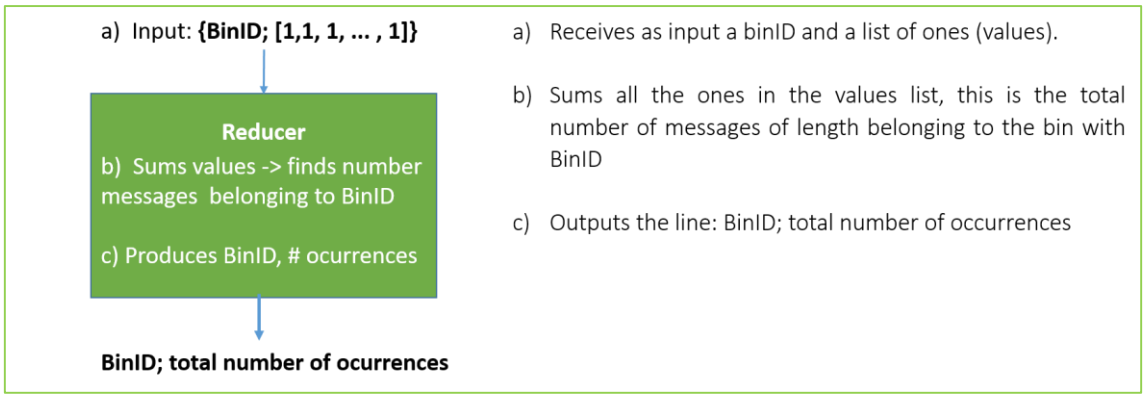
To implement this the keys produced in the mappers were changed to be bin identifiers.

For each tweet arriving at the mappers the length is computed as well as the bin to which this message belongs to.

Each message produces a pair (BinID, 1). The reducer part stays the same as in WordCount, i.e. for each BinId it will sum up all the values and find the total number of occurrences. This means that for each bin we will have the number of messages which length are equal to a length that that bin represents.

A schematic explanation of the mappers and Reducers functions is presented below:





- Data cleaning, parsing and specific coding choices…

Each input had to be processed in order to identify the tweet message and compute its length. First the input is transformed to a string by using the `toString()` method from the `Text` class. This `String` is stored in the variable `line`.

Then, Since we know the structure of the input (four fields separated by semi-colon) `line` is separated into substrings delimited by semi-colon, with `split(";")` from the `String` class. This method returns a `String` vector with each element as a substring. This vector is stored in the `fields` variable.

The first validation of the input is performed at this point. If the number of fields found is different than 4 then this input is discarded.

The number of fields may be different of 4 for different reasons:

a) The tweet may have not been well captured from the stream
b) The message itself has some semi-colon inside.

This Tweets could be corrected instead of discarded, or the percentage of tweets with these issues could be computed to get a better idea of the impact of these action. This inputs were discarded as a choice regarding the scope of this module, but this limitation is identified.

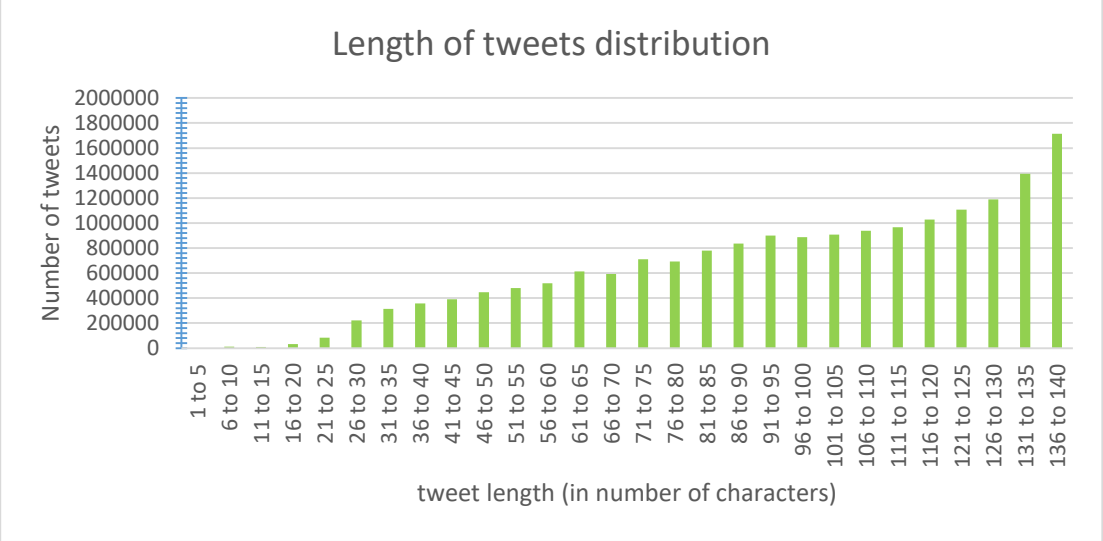
The tweet message is expected to be the 3rd field of the input which is written to the variable `tweet`. The message length is computed by the String method `length()` and this length is then used to compute the `BinID`.

Twitter had, in the time of Rio2016, a limit to the message length of 140 characters. Some languages however use special characters that are identified as more than 1 character when captured from Twitter. These messages when processed in this program could result in computed lengths higher than 140 characters. These tweets were disregarded. Since, even if considered, these tweets would fall in the bins of larger lengths, filtering them out does not alter the shape of the distribution found in the results.

`BinID` is of IntWritable type with values from 0 to 27 representing each final bin that groups together the counts of messages with lengths in five characters intervals. This is computed by `(tweet.length() -1) / 5` where `/` is the integer division. The `-1` term adjusts the boundary values to get the desired bins, for instance: 5 characters length belongs to 1st bin (BinID=0), but 6 characters length belongs to the 2nd bin (BinID=1).

- Results and analysis

For readability purposes the BinId is translated to the length interval it represents.



The first aspect of notice is that the first bin has zero messages, meaning that not a single message was found with only 1 to 5 characters. This is due to the fact that these tweets were selected by having its content in some way related to the Rio2016 games, in specific the text had to include either one of this hashtags #Rio2016 or #rioolympic which will make the message length at least 8 characters long.

The most frequent length is close to the limit 140characters which shows a general trend to use the whole available space for the message.

Since the tweets with special characters were not excluded, the results are biased to higher lengths since these messages appear with higher lengths than the real.

B - Twitter Time Analysis:

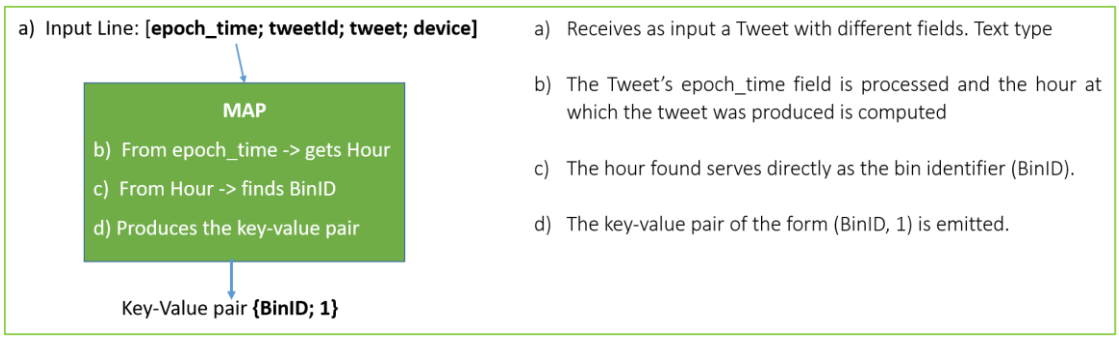
Goal: Create a histogram representing the distribution of number of tweets over the 24 hours of a day.

- Map-reduce problem:

The same general idea applied before is used here. The key to be passed to the reducers and around which the values will be aggregated is the final bin of the histogram we want to create. In this problem each bin represents one of the 24 hours of the day.

The processing of the input to find the key is done in the mappers which produce a key value pair of the form (Hour, 1) for each tweet in the input.

Since the idea is to count the number of occurrences of a key, i.e. number of tweets sent at each hour, the reducer part for this problem will stay exactly the same as before. Outputting a pair in the form (Hour, total number of tweets)



- ## Data cleaning, parsing and specific coding choices.

Each Tweet of the twitter database is parsed as before and the timestamp of each tweet is stored in the 1st element of the `fields` array.
This timestamp is in the epoch format in miliseconds. This string is used as argument to create an object (`tweetTimeStamp`) of the type `Date`. The hour can be retrieved using the `Date` class method `gethour()`. The value obtained is an integer between 0 and 23 that corresponds to the hour in UTC.

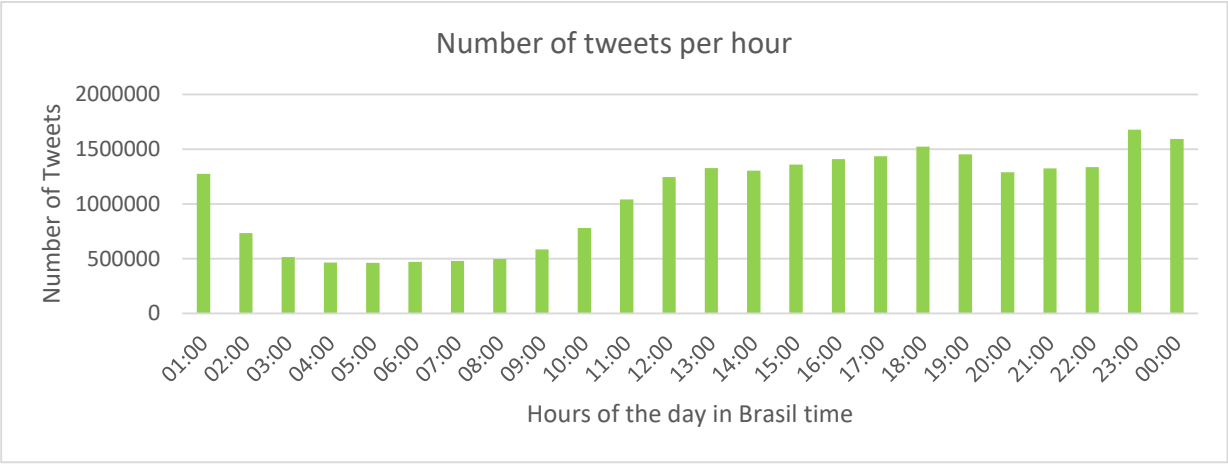
To validate the input tweets with the following characteristics were disregarded:

a) Number of fields different than four;
b) The first element of fields not numerical.

In order to be able to relate the times at which the tweets were sent with the actual events occurring at the games, an additional filter was created to consider only tweets sent during the Olympic Games' days. By using the class `Date` methods `getmonth()` and `getday()` `Tweets` which date falls outside these days are disregarded. Both versions, with and without this filter were ran.

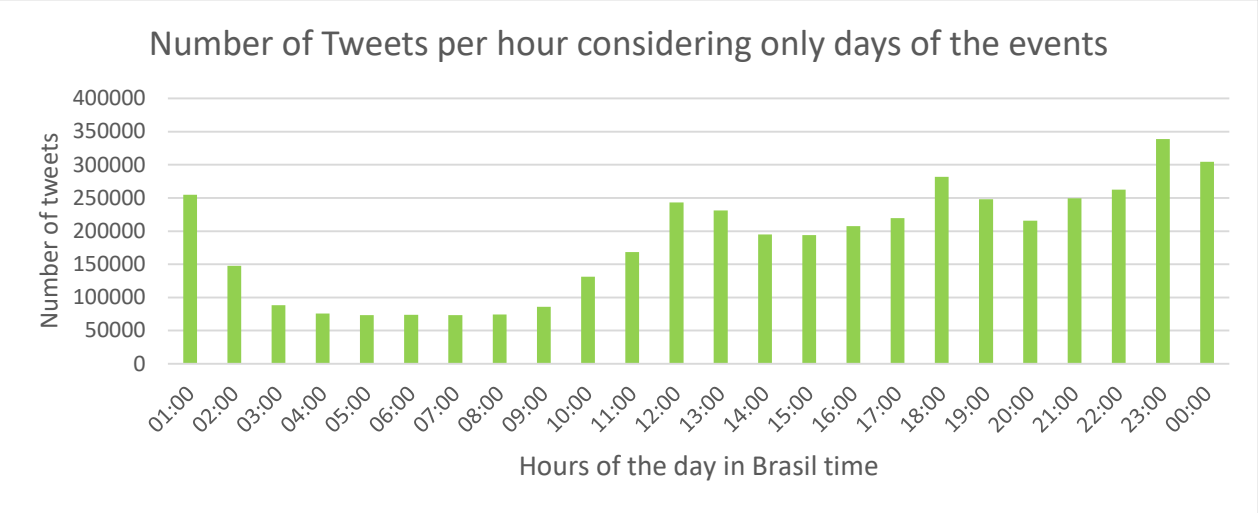
- ## Results and analysis

To be able to relate the results with the events occurring in Olympic Games, the histograms have the hours translated from UTC to Rio de Janeiro's local time. The difference to UTC at the time of the games was of -3 hours.
The following histogram includes tweets from all days in the dataset.



The busiest hour was 11pm Rio de Janeiro time. Overall the periods with more tweeting activity are around lunch time (12pm to 1pm), later in the afternoon (5pm to 7pm) and later in the night until 1am. There is a significant slower period during the first hours of the day.
Regarding these results it is not certain whether these tendencies are related to the events occurring during the Olympic Games or if in fact they translate a common tweeting behavior.

To understand this, the program was ran again considering only the tweets produced during the days of the events: 5 to 21 of August. The number of tweets produced during this period is only 17% of the total number of tweets considered before.



Looking at this distribution, although the same tendency as before can be seen, it is now better identifiable short periods of great activity, namely at 12pm, 6pm and 11pm.

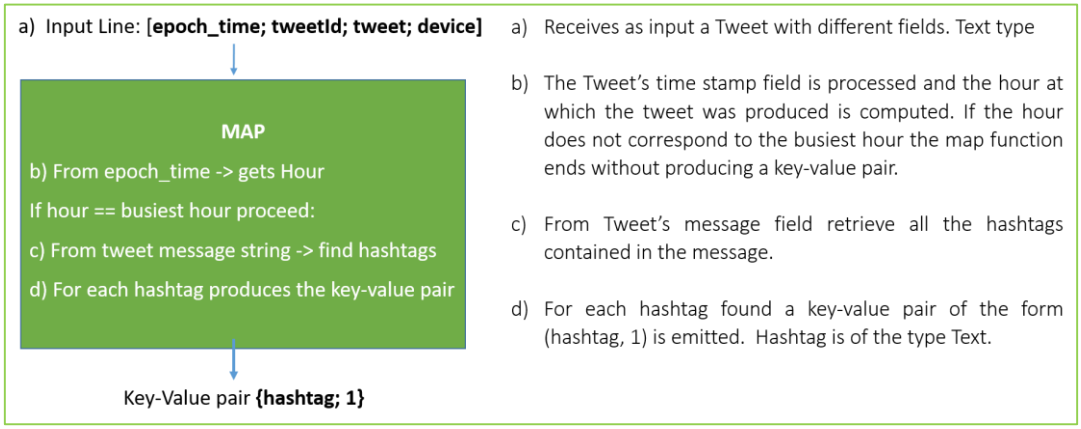
These periods are used, together with the hashtags to identify what was happening at the Rio2016 games.

B2-Twitter Hashtags Analysis

Goal: Identify the top 10 hashtags used during the busiest hour.

- Map-reduce problem:

The problem can be seen again as a count of occurrences of a certain feature, where in this case the feature is a hashtag.
In the Mappers, The hashtags are retrieved from the tweets' message field and each different hashtag will produce a key-value pair (hashtag, 1).
The Reducers sum all the occurrences of unique hashtags and output a list of unique hashtags and their total number of occurrences.



- Data cleaning, parsing and specific coding choices

Each tweet input reaching the mappers is processed in the same way as before in order to get the hour at which the tweet was produced. Here the only tweets from which we want to extract hashtags are the ones produced at the busiest hour found in the previous question, thus disregarding all other tweets. Having found a tweet produced at this hour, this tweet's message is retrieved by assessing the 3rd element of the `fields` array.

Twitter hashtags are not case sensitive, therefor, the message string is transformed to lower case. Hashtags were identified with the following regex expression `[\\s\\W]#(\\w+)` .

This expression encodes the main aspects of the Twitter hashtag:

a) Captures an alphanumeric word (numbers, letters and underscores) after a hash symbol;
b) Only allows white spaces and other non-alphanumeric symbols before the hash symbol.

This elegant expression fails to capture a hashtag at the very beginning of the message. To overcome this edge case a space is added to the beginning of the message string before running it through regex.

The method co`mpile()` from the class `Pattern` was used with this regex to define the pattern to look for, then the method `matcher()` of the same class is applied to the message string and returns the matches found in an object of type `Matcher`. This object contains each hashtag and a key-value pair is emitted for each.

The filters regarding the number of fields and the validation of the first field as a numeric string are here applied also in order to clean "bad" tweets. Regarding the date of the tweets, the same additional filter was also applied.

- Results and analysis

The program was ran once considering all the tweets from the 23rd hour (2am UTC) from all days in the dataset.

Table 1 shows the top 10 hashtags found in increasing order of occurrence:

Hashtags	Count
openingceremony	35882
cerimoniadeabertura	36488
swimming	36636
oro	40895
usa	42738
futebol	49357
bra	50250
gold	68119
olympics	91735
rio2016	1448262

Table 1- top10 Hashtags for 11pm using all tweets

The most frequent used hashtags at this hour were #olympics and #rio2016. This is expected since these hashtags were used as indicators that the content of the tweet was related to the Rio2016 games and thus as a selection criteria for the tweet to be included in the dataset. The other hashtags seem to be of popular subjects like #BRA for Brasil, #gold and #oro for the gold medals. #futebol, #swimming, #cerimoniadeabertura and #opencerimony may be related to the events occurring in the games at this hour. To better evaluate this, the program was ran again using the same hour but applying the filter to select only tweets from the days of the events. The results are showed in table 2.

Considering only Tweets produced at 11pm during the days of the events, the most used hashtags are related to de Opening Ceremony. And indeed, looking to the Rio 2016 official schedule, this ceremony started at 8 pm and finished around 11pm, which is in agreement to this being a popular topic at that time.

Top 10 Hashtag:	Count
juegosolimpicos	6789
swimming	7819
teamusa	8478
usa	10683
oro	11180
gold	14537
olympics	20772
openingceremony	32881
cerimoniadeabertura	33463
rio2016	292506

Table 2- top10 hashtags at 11pm from Tweets during the period of the Olympic

Since it was identified in the previous question that the hours 12pm and 6pm were hours of increased activity, it was interesting to try to match the top hashtags with events happening at that times as well.

Top 10 Hashtag:	Count
juegosolimpicos	7727
gold	9052
oro	9802
judo	11342
olympics	12578
tenis	12763
bra	13895
arg	16641
futebol	16674
rio2016	245519

Table 3- top10 hashtags at 18pm in tweets in days of event only

At 12 pm the second most popular hashtags was #badminton however a clear relation between the badminton games schedule and the 12pm hour couldn't be found. Although when considering the hashstags for tweets produced at 6pm (see table 3), it was evident the most talked about event. Looking at the football games schedule, at exactly this hour Argentina played against Algeria in 7th of August. Likewise the man's football's final match was between Brasil and Germany also played at this hour in the 20th of August.

C - Athletes Public Support Analysis

Goal: Find the 30 athletes who got more mentions in Twitter.

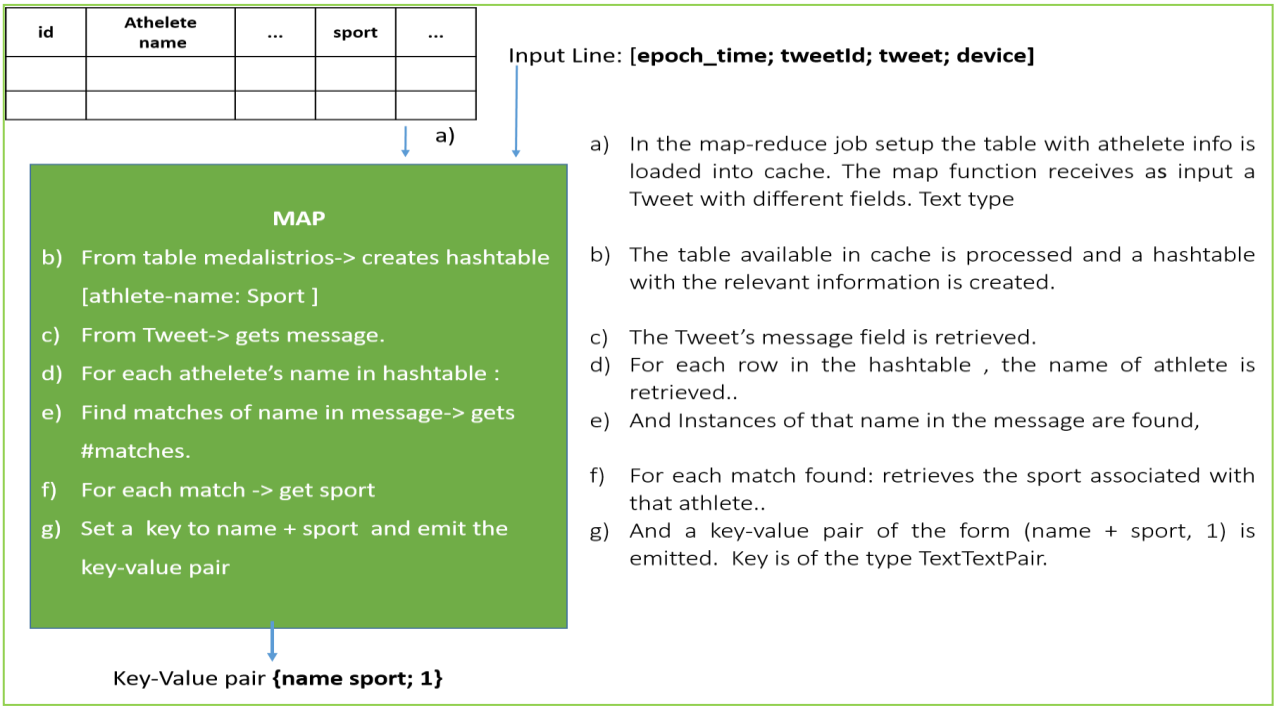
- Map-reduce problem:

Here, the objective is to count the number of times an athlete's name appears in the messages of Twitter. The map-reduce structure will basically be the same since from the moment an athlete name is found in the text, the key is set using that athlete's name and a pair key-value is emitted with value 1. The reducer sums the values up for each unique athlete and in the end a list with total number of mentions for each athlete is produced.

In order to know the names to look for in the tweet's messages a second database is used (medalistrio.csv). During the Job Setup this database is loaded into cache memory, each mapper starts by reading the information needed from this table into a hashtable.

From this table two fields are selected and passed to the hashtable: Athletes' names and Sport. The Sport of each athlete is selected in order to facilitate the map-reduce job for the next question and this information is passed inside the Key together with the athlete's name. The key for this job has two parts: athlete name and Sport. It is created by using a new type called TextTextPair, which was developed following the available example of IntIntPair .

The possibility that the introduction of the sport in the key could cause an athlete name to be considered in multiple keys was considered. This could happen if some athlete would play different sports, but inspecting the medalistrio table this situation was not identified. For now apart from the key type difference, the sport information can be ignored since it won't change in any way how this program is structured.



- Data cleaning, parsing and specific coding choices

The hashtable is created during the Map Setup. Here the cached table with the athletes' information is assessed and row by row the content is parsed into fields using the delimiter ",". The hashtable *athletesinfo* is filled with the 2nd and the 7th fields read that correspond to an athlete's name and her sport.

The mapper function begins by parsing the input tweet into fields and the message field is transformed to lower case.

For each entry of the hashtable a string name is created by using the proper methods from the *Hashtable* class. This string name will also be transformed to lower case. This is done in order to be able to find more mentions to an athlete's name by allowing a more flexible match.

The name in lowercase is therefore used as the substring pattern to be found in the message string. By using the *countmatches(message, name)* method of the *StringUtils* class, the total number of matches is obtained.

For each match, the sport associated to the name in the hashtable is also retrieved by the method *get(key)* from the hashtable class. Having the athlete's name and sport the map-reduce key is set by using the *set(string1, string2)* method of the *TextTextPair* class created. After a key value pair (athlete name sport ; 1) is emitted.

The program could be made more efficient if the sport was retrieved only once per name and not for match. Also, instead of emitting a pair for each match, a single emition could be done for one athlete'sname and by modifying the value emitted from 1 to the number of matches found for that name. For this the reducer would have to be changed to sum the values instead of just increment a counter. The results would be the same but the number of pairs emitted would be less. Considering the small time it took to run, these changes were not considered important.

The results are shown and discussed together with the results for the next question (table 4).

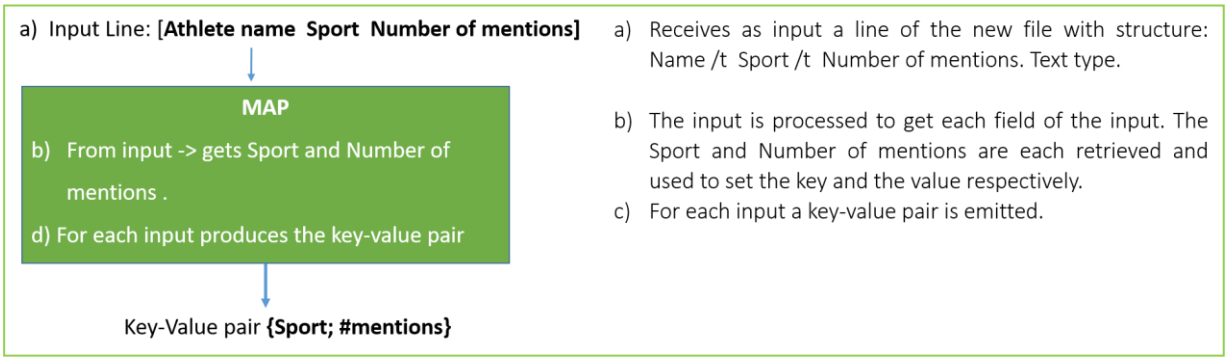
C2 - Sports Popularity Analysis

Goal: Based on the number of mentions per athlete find the top 20 most popular sports.

- Map-reduce problem:

By using as input the output from the previous question, a simple aggregation of the number of mentions around Sport is able to find the most popular sports. To perform this each line of the input, referring to a single athlete, needs to emit as key the sport and as value the number of mentions of that athlete. (Sport, 1)

The reducer sums all this mentions for the same sport and produces a list with Sport-> total number of mentions of athletes of that sport.



- Data cleaning, parsing and specific coding choices

The file used as input is structured in fields [Name, sport, Number of mentions.] and each line addresses one single athlete.

A mapper receives one line of this file in a type Text variable. In a similar way when processing the Tweet input, this line is parsed to obtain the different fields separated by tabs (`/t`).

The 2nd field is retrieved and its value is used to set the key Sport. The Value part of the Key-Value pair to be emitted is the number of mentions of each athlete. This number is obtained from the 3rd field of the input and needs to be translated to an integer. This is done by using the `parseint()` from the `Integer` class. This integer is finally used to directly set the value of the Value to be emitted.

- Results and analysis

The 30 athletes most mentioned in the twitter database are presented in table 4, in decreasing order.

The most popular athletes found are the Swimmer Michael Phelps, followed by the Brazilian Neymar, and in 3rd position the runner Usain Bolt. All this are generally well known athletes even outside the sports worlds, so it makes sense to find them in the top 3 positions.

Regarding the top 20 Sports, the results are presented in table 5 in decreasing order.

It is interesting to find that the top3 sports correspond to the sports of the most mentioned athletes.

Football, being a very popular sport worldwide makes sense to be in the top 3 of the sports list, specially since so many football players are well known which contributes to the number of mentions to be high.

Athletic, Aquatics and Gymnastics should be seen as groups of sports with many variations and events associated, these contributes to a higher number of athletes in this groups therefore contributes to a higher total number of mentions.

Athlete	Sport	# Mentions
Michael Phelps	aquatics	191332
Neymar	football	185099
Usain Bolt	athletics	176774
Simone Biles	gymnastics	82932
William	football	67692
Ryan Lochte	aquatics	41823
Katie Ledecky	aquatics	40265
Yulimar Rojas	athletics	35667
Simone Manuel	aquatics	28433
Joseph Schooling	aquatics	26997
Rafaela Silva	judo	26076
Sakshi Malik	wrestling	25282
Wayde van Niekerk	athletics	23059
Andy Murray	tennis	22367
Kevin Durant	basketball	21525
Tontowi Ahmad	badminton	21153
Liliyana Natsir	badminton	20498
Monica Puig	tennis	18720
Andre de Grasse	athletics	18218
Penny Oleksiak	aquatics	18118
Rafael Nadal	tennis	16830
Laura Trott	cycling	16386
Luan	football	16038
Ruth Beitia	athletics	15942
Teddy Riner	judo	14742
Lilly King	aquatics	14508
Jason Kenny	cycling	12574
Shaunae Miller	athletics	12404
Elaine Thompson	athletics	12293
Caster Semenya	athletics	12055

Tabela 4- top30 atheletes by number of mentions in the Tweets. In decreasing order

top 20 SPORTs	# mentions
athletics	496967
aquatics	472683
football	322099
gymnastics	138948
judo	105334
tennis	89263
basketball	77009
cycling	71565
badminton	63813
wrestling	35864
weightlifting	26034
sailing	25076
canoe	24582
shooting	24376
equestrian	24083
boxing	23804
volleyball	18084
rowing	17938
taekwondo	16383
fencing	13777

Table 5- top20 Sports by number of mentions of their Athletes. In decreasing