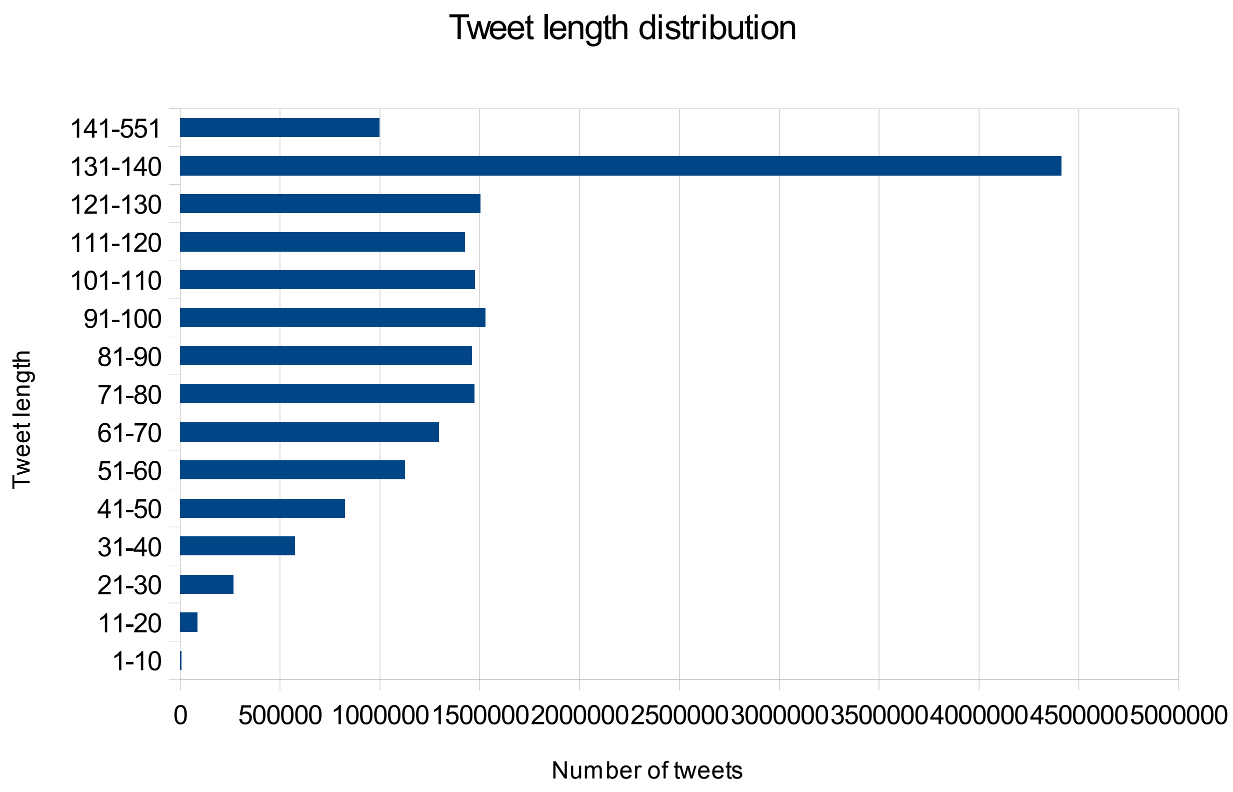
Part A. Text Analysis

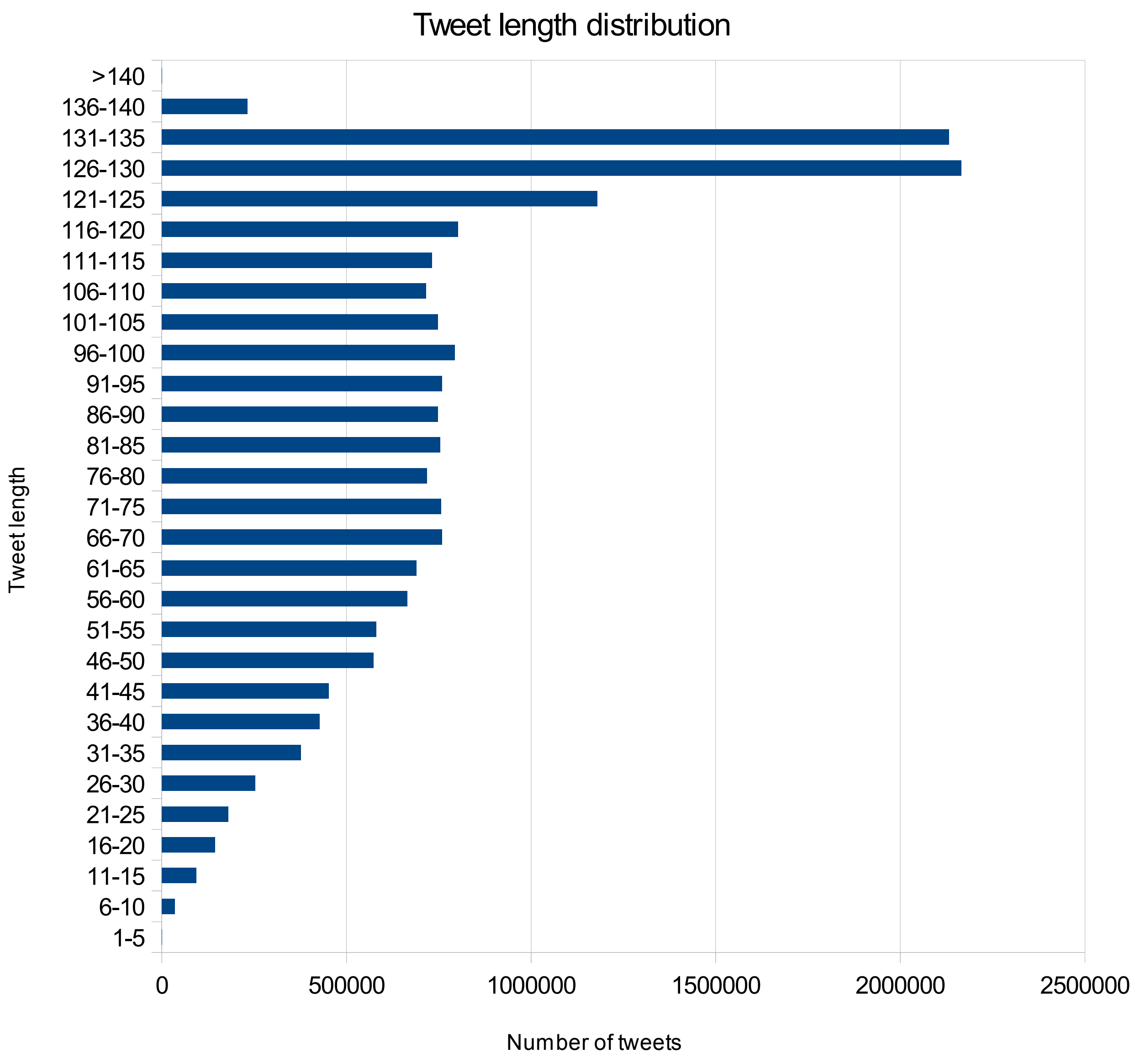
For this part of the assignment I implemented two different Hadoop jobs. First one – TextAnalysis – retrieves the distribution of tweet lengths; and second one – TextAnalysisAvg – is responsible for calculating the average tweet length.

TextAnalysis

First, I implemented a mapper class – TMapper, which emits key-value pairs in a form of *(tweet message length, 1)*, where each tweet message is the last field of the tweet data string (retrieved with StringUtils). This allows to group values by tweet length and then sum up all the occurrences of tweets of a certain length, which is done in TReducer. Then, I ordered the keys and grouped them by 10 using Excel to produce a graph presented below. As there are values that exceed the allowed tweet length (140), I assumed that any results over 140 come from the tweets in foreign languages or containing some special characters.



From the graph above it can be seen that the number of tweets that are over 140 characters long, is too significant to be ignored, which is why I decided to alter my code in order to do more accurate filtering. I replaced all non-alphanumeric characters in each tweet by "1" with a use of regular expressions for alphabetic and numeric characters, so that the tweet length would be calculated more precisely. There were still 481 tweets whose length was over 140 characters, which I assume comes from links included in tweet messages, but this number is almost negligible comparing to the overall lengths distribution. Apart from that, I implemented grouping by 5 in my mapper in order to avoid doing it manually in Excel. Finally, I added an if-else statement that enables a “no-care” mode for tweets that have over 140 characters and groups all of them together in the reducer – *(>140, 1)*. I emitted each key-value pair in a form of *(lower-bound length– upper-bound length, 1)* for all tweets below 140 characters, which was done by dividing each tweet length by 5, rounding it to a higher of the 2 closest integers, and then multiplying it back by 5. I also calculated the lower bound which was done in relatively similar manner, but which less than the upper-bound by 4 . All these updates eliminated any need for manual work in Excel and the only thing that I needed to do was to compose a graph from the output data, which is provided below. Even though there are significant changes in the graph, the overall shape of the curve (except for oversized tweets) still remains the same for this dataset, be it with or without special characters.

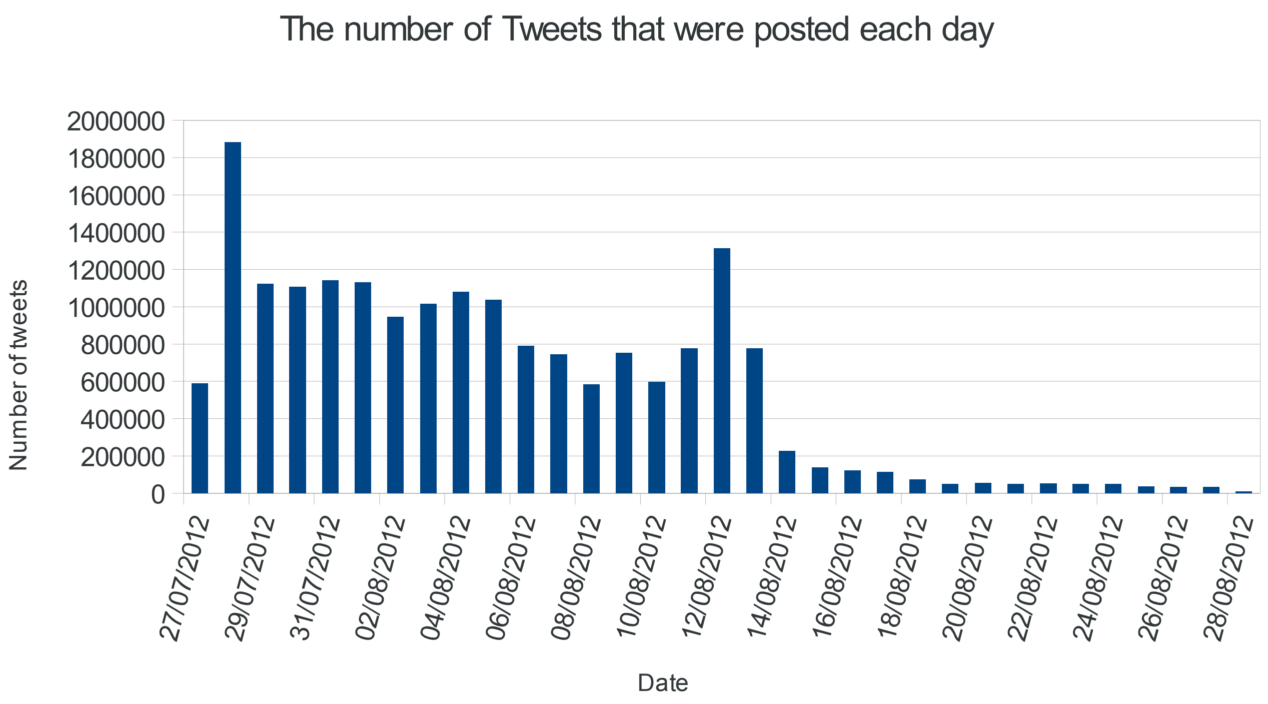


TextAnalysisAvg

Within the mapper class – TAvgMapper I used IntIntPair class to form aggregated values in order to emit key-value pairs in a form of *("avg", (tweet message length, 1))*. I specifically used a common "avg" key for all the values in order to combine them in the reducer class – TAvgReducer, in which sum of tweet message lengths was divided by sum of the occurrences of these lengths in order to achieve an average. My Hadoop job produced 100 as its result, which I then checked in Excel where the average tweet length was also roughly equal to 100 (100.45). I then altered my source code to get more accurate results by adding the same non-alphanumeric character replacement mechanism that I used in TextAnalysis, which resulted in the average of 92. This seems to be completely logical, as there were a lot of tweets with length over 140 characters, which are now much shorter due to filtering out special characters, and that is why the average gets more precise and even more balanced now.

Part B. Time Analysis

For part B as a mapper class I implemented TimeMapper and as a reducer class – TimeReducer. In TimeMapper first, I retrieved the tweet date string (using StringUtils). As for this task we only needed day, month and year, I partitioned the date string by commas and used only first part as a key value. From my mapper class I was emitting key-value pairs in a form of *(tweet day-month-year, 1).* Then, in TimeReducer, I summed up all occurrences of each date and from the output of the Hadoop job I did compose a graph in Excel, which is presented below.



Part C. Hashtag Analysis

Most of the work for this part was done in the HMapper. I did pattern matching for hashtags (substrings starting with "#") within the tweet messages, as when I tried to search among the tags themselves in a hashtag field, I encountered words containing "go" or "team" not only in the start of the tag, but also in the middle or in any part of the word. I specifically matched patterns starting with "#go" and "#team" by regular expressions in order to return all substrings of the tweet messages starting with "#go" or "#team" and ending with a whitespace. In order to group properly the teams, prefixes "# go" and "#team" were stripped off from the key values and key values were converted to lowercase strings, in order to obtain a cumulative number for hashtags for both of them. Moreover, in order to cover hashtags that look like"#goteam", and to combine the values properly, I implemented a nested pattern matching against "#go" and then "team" in order to retrieve the actual name of the supported team. I decided to include the "#goteam" processing, as there was a significant number of tags in the output result starting with this prefix.

Within the HReducer I simply added the number of hashtags encountered for each team.

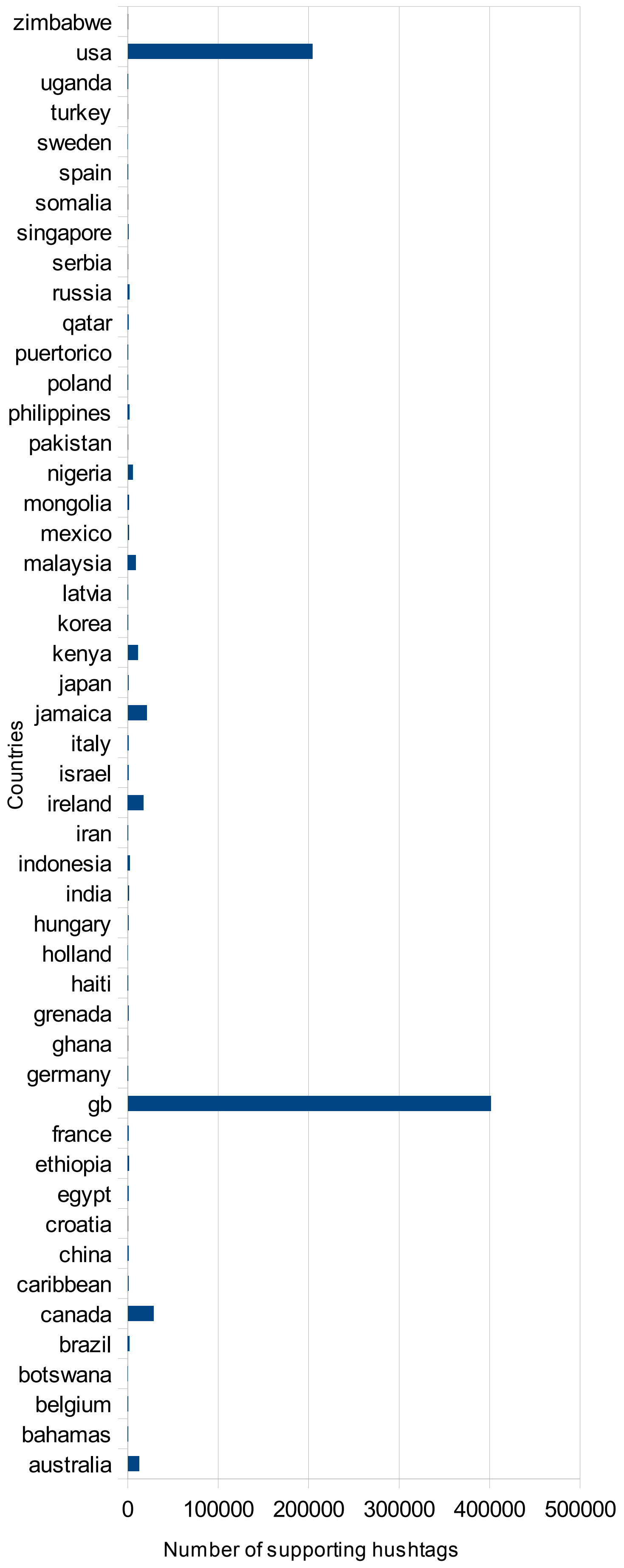
For some countries people misspelled country names, used slang names or added extra information in the end. For instance, for Australia's team people used "aus", "aussie", "aussies" and "australia", whereas for Canada's team people tended to write "cana", "canad", "canadago" or the most popular – "canada". As it seems too difficult to filter this automatically and accumulate the values in hadoop cluster, this was done manually in Excel. I also had to filter out the teams that did not resemble any country names (for example, "brijamada"). However, this was not that hard considering that the output text file from hadoop was manageable enough.

Even though there was an option to filter out the numbers of hashtags, outputting only those that are exceeding some specified minimum to make the output shorter, I decided not to take this approach, as I did not want to risk loosing some valuable data from the tweets dataset. Apart from that when I tried to limit hashtag numbers to start only from 100, it did not change significantly the output size.

The table and graph presented below illustrate a list of countries against the number of supporting hashtags for each of them in the tweets dataset. It can be seen both through the graph and the table that the largest number of supporting tweets were received by gb team, followed by usa, whereas some countries have almost negligible number of supporting hashtags – for example, Zimbabwe.

Teams	Hashtags #
australia	12609
bahamas	559
belgium	505
botswana	227
brazil	2190
canada	28719
caribbean	807
china	1177
croatia	431
egypt	1107
ethiopia	1651
france	1224
gb	401203
germany	532
ghana	426
grenada	945
haiti	740
holland	206
hungary	827
india	1434
indonesia	2646
iran	570
ireland	17798
israel	1048
italy	1052
jamaica	21033
japan	985
kenya	11152
korea	515
latvia	582
malaysia	9077
mexico	1355
mongolia	1673
nigeria	5716
pakistan	475
philippines	2159
poland	575
puertorico	753
qatar	1077
russia	2093
serbia	303
singapore	785
somalia	324
spain	550
sweden	206
turkey	391
uganda	728
usa	204269
zimbabwe	286

Support for teams through hashtag analysis


zimbabwe
usa
uganda
turkey
sweden
spain
somalia
singapore
serbia
russia
qatar
puertorico
poland
philippines
pakistan
nigeria
mongolia
mexico
malaysia
latvia
korea
kenya
japan
jamaica
italy
israel
ireland
iran
indonesia
india
hungary
holland
haiti
grenada
ghana
germany
gb
france
ethiopia
egypt
croatia
china
caribbean
canada
brazil
botswana
belgium
bahamas
australia
Countries
Number of supporting hushtags