Assignment 5

Versions of frameworks

Scala: 2.11.0

Spark: 2.2.1

Running of the program

The jar file triggers a stream of tweets with the twitter API and does the requested analysis on them. In order to run the file the following commands should be used:

spark-submit –class TwitterStreaming Utkarsh_Gera_hw5.jar

spark-submit –class BloomFiltering Utkarsh_Gera_hw5.jar

As mentioned in the problem description the program prints the output on the console for both the tasks. The first task executes the Reservoir Sampling algorithm by limiting the numbered of the stored tweets to 100. Then based on the probability of 100/n for the n^{th} tweet, it either picks discards or picks up the n^{th} tweet and randomly replaces any tweet from the sample with it. It does this for the number of tweets chunks supplied by the twitter API after every 10 seconds. Since the stream is never ending, the program has to be killed manually.
It prints the tweet number, that is going to be placed in the sample, from the beginning of the stream along with the top 5 hot hashtags and the average tweet length (in terms of the number of characters).

```
The number of the twitter from beginning: 160
Top 5 hot hashtags:
ShandurPoloFestival:3
KP:3
EXO:2
geelong:2
HowtoPerfect:2
The average length of the twitter is: 122.3


The number of the twitter from beginning: 164
Top 5 hot hashtags:
ShandurPoloFestival:3
KP:3
EXO:2
geelong:2
HowtoPerfect:2
The average length of the twitter is: 122.37
```

The second task was implementing the bloom filtering on the hash tags encountered on the twitter input stream tweets. Like the first task the second task also receives a chunk of tweets after every 10 seconds from the input stream. We filter out the hash tags we have seen using the two hash functions modulo by 64 and 59 and the size of the filter is 64 filters/bits. As mentioned in problem statement, program prints number of correct as well as incorrect estimation(false positives) for that chunk of tweets, that it receives after every 10 seconds. It also prints global statistics: number of unique hash tags and number of False positives by the Bloom Filtering algorithm encountered till that chunk since the beginning of the stream (Reference to my post 116 in the board). If we increase the bloom filter size to something greater than 64 bits/filters or if we introduce a way to handle negative hash codes for the tags(currently the program considers only absolute value) which will decrease the collisions or both will lead to increase in accuracy.

```
> Local statistics for this chunk -
Number of Correct estimation in this chunk: 50
Number of Incorrect estimation/False Positives in this chunk: 42
> Global statistics from starting till this chunk -
Number of actual unique hash tags encountered up till now: 86
Number of False Positives by the bloom filter up till now: 42
----------------------- End of this chunk -----------------------
```

P.S. The names of the source code files have been modified as mentioned in the description to “Firstname_Lastname_TwitterStreaming.scala” or “Firstname_Lastname_BloomFiltering.scala”, but the class name in the code remains TwitterStreaming and BloomFiltering.