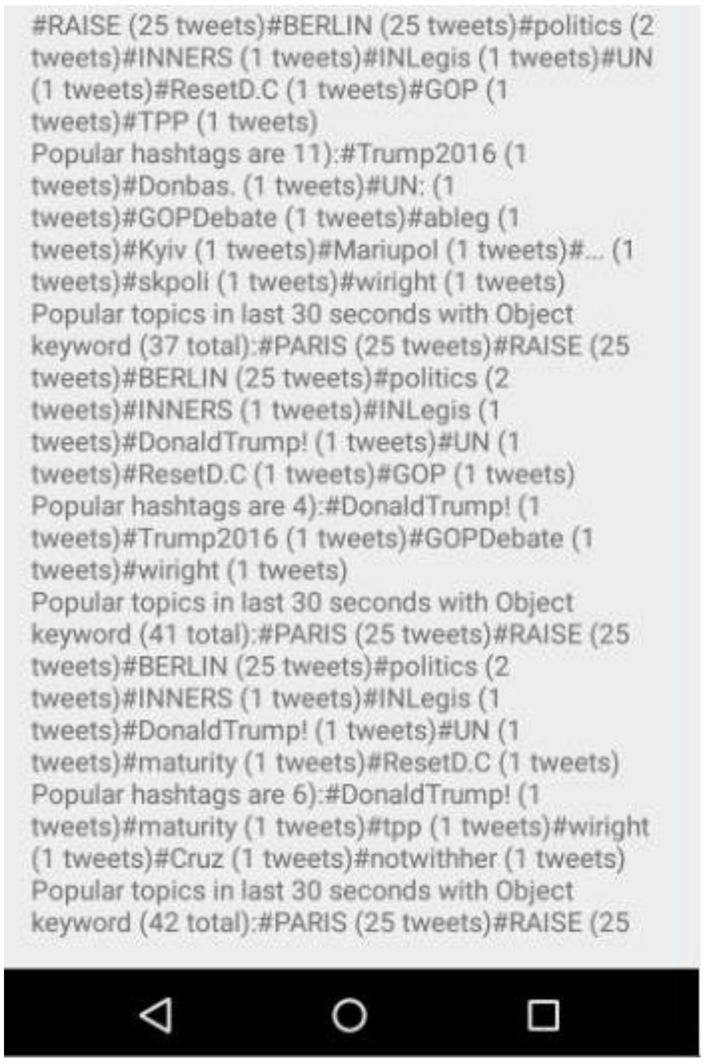
Report:

1.

Collecting the twitter streaming data and performing word count on the collected data and sending back to the android with the word count of analyzed twitter data.

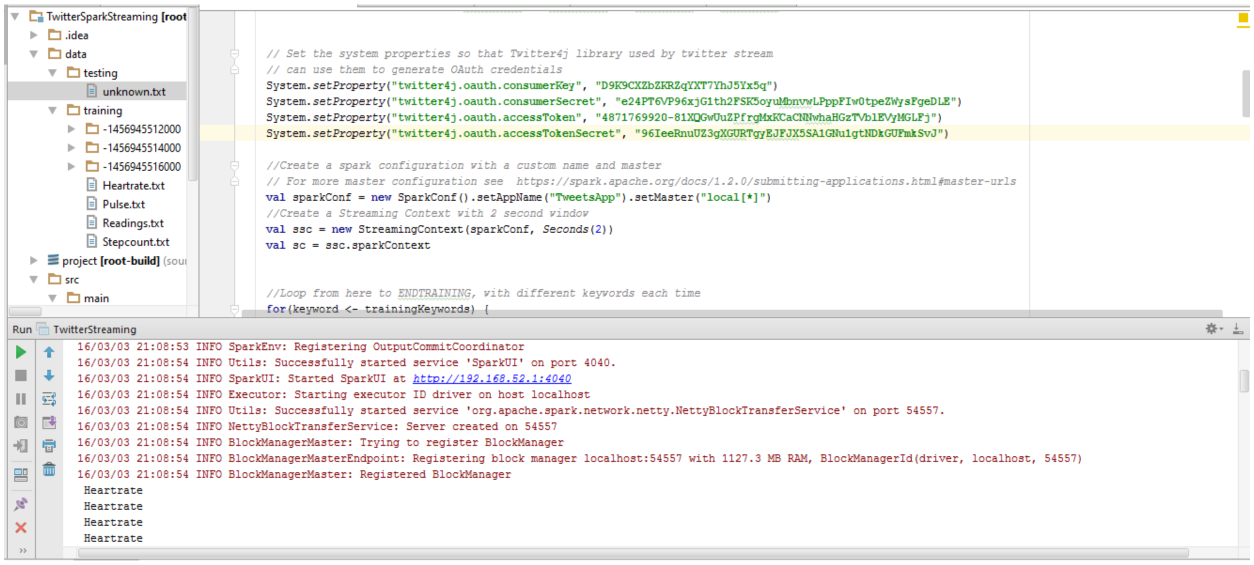


```
#RAISE (25 tweets)#BERLIN (25 tweets)#politics (2
tweets)#INNERS (1 tweets)#INLegis (1 tweets)#UN
(1 tweets)#ResetD.C (1 tweets)#GOP (1
tweets)#TPP (1 tweets)
Popular hashtags are 11):#Trump2016 (1
tweets)#Donbas. (1 tweets)#UN: (1
tweets)#GOPDebate (1 tweets)#ableg (1
tweets)#Kyiv (1 tweets)#Mariupol (1 tweets)#... (1
tweets)#skpoli (1 tweets)#wiright (1 tweets)
Popular topics in last 30 seconds with Object
keyword (37 total):#PARIS (25 tweets)#RAISE (25
tweets)#BERLIN (25 tweets)#politics (2
tweets)#INNERS (1 tweets)#INLegis (1
tweets)#DonaldTrump! (1 tweets)#UN (1
tweets)#ResetD.C (1 tweets)#GOP (1 tweets)
Popular hashtags are 4):#DonaldTrump! (1
tweets)#Trump2016 (1 tweets)#GOPDebate (1
tweets)#wiright (1 tweets)
Popular topics in last 30 seconds with Object
keyword (41 total):#PARIS (25 tweets)#RAISE (25
tweets)#BERLIN (25 tweets)#politics (2
tweets)#INNERS (1 tweets)#INLegis (1
tweets)#DonaldTrump! (1 tweets)#UN (1
tweets)#maturity (1 tweets)#ResetD.C (1 tweets)
Popular hashtags are 6):#DonaldTrump! (1
tweets)#maturity (1 tweets)#tpp (1 tweets)#wiright
(1 tweets)#Cruz (1 tweets)#notwithher (1 tweets)
Popular topics in last 30 seconds with Object
keyword (42 total):#PARIS (25 tweets)#RAISE (25
```

2.

Test Data: collecting different kind of twitter data related to the heartrate and generating a model

Train data: Upcoming twitter streaming data .



```
// Set the system properties so that Twitter4j library used by twitter stream
// can use them to generate OAuth credentials
System.setProperty("twitter4j.oauth.consumerKey", "D9K9CXZbZKRZqYXT7YhJ5Yx5q")
System.setProperty("twitter4j.oauth.consumerSecret", "e24PT6VP96xjG1th2FSK5oyuMbnvwLPppFIw0tpeZWysFgeDLE")
System.setProperty("twitter4j.oauth.accessToken", "4871769920-81XQGwUuZPfrgMxKCaCNNwhaHGzTVblEVyMGLFj")
System.setProperty("twitter4j.oauth.accessTokenSecret", "96IeeRnuUZ3gXGURTgyEJFJX5SA1GNu1gtNDkGUFmkSvJ")

//Create a spark configuration with a custom name and master
// For more master configuration see  https://spark.apache.org/docs/1.2.0/submitting-applications.html#master-urls
val sparkConf = new SparkConf().setAppName("TweetsApp").setMaster("local[*]")
//Create a Streaming Context with 2 second window
val ssc = new StreamingContext(sparkConf, Seconds(2))
val sc = ssc.sparkContext


//Loop from here to ENDTRAINING, with different keywords each time
for(keyword <- trainingKeywords) {
```

```
16/03/03 21:08:53 INFO SparkEnv: Registering OutputCommitCoordinator
16/03/03 21:08:54 INFO Utils: Successfully started service 'SparkUI' on port 4040.
16/03/03 21:08:54 INFO SparkUI: Started SparkUI at http://192.168.52.1:4040
16/03/03 21:08:54 INFO Executor: Starting executor ID driver on host localhost
16/03/03 21:08:54 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 54557.
16/03/03 21:08:54 INFO NettyBlockTransferService: Server created on 54557
16/03/03 21:08:54 INFO BlockManagerMaster: Trying to register BlockManager
16/03/03 21:08:54 INFO BlockManagerMasterEndpoint: Registering block manager localhost:54557 with 1127.3 MB RAM, BlockManagerId(driver, localhost, 54557)
16/03/03 21:08:54 INFO BlockManagerMaster: Registered BlockManager
 Heartrate
 Heartrate
 Heartrate
 Heartrate
```