**CS560 - Knowledge Discovery and Management**

**Lab4 – Assignment**

**Problem Statement:**

Complete the following tasks:   
1. Cloudera/Mahout/Solr: Implement a Mahout application (e.g., Classification/Recommedation/Association Rule Mining) using your own data and store your output to Solr.  
2. Mobile Application: Implement a mobile application to use the information through Solr REST Web Services  
3. Write a short report on your work (including screenshots).   
4. Post all your work (source code and report) to Lab 4 directory of your GitHub site. And post your GitHub Lab 4 link to the following site

**1)Classification:**

Used Classification Algorithm similar to Twitter on my own data set of sensor data.

Steps to train the matrix explained in short report in point (3).

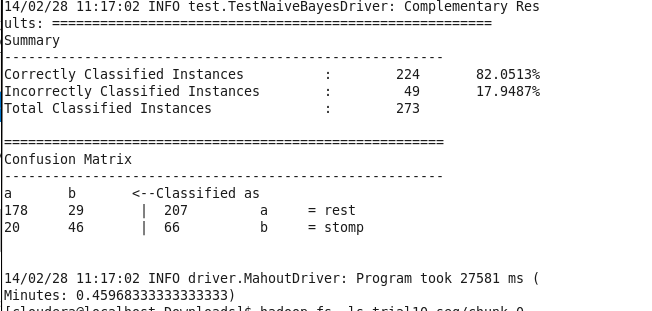
Trained the data and the classification matrix after test data set:

82% accuracy.

Data set: 700

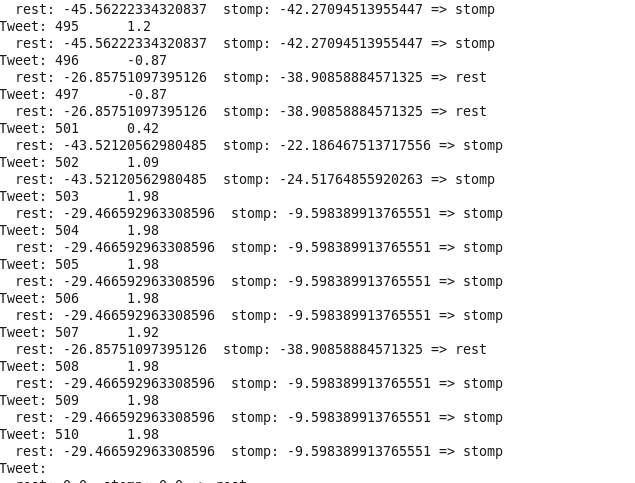
Train data set : 427

Test data set: 273

****

Classified the “to-classify”(new set of sensor data with axis vales):

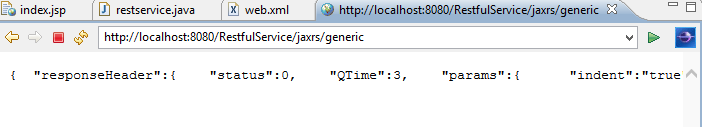
Rest and stomp is classified. Used the existing class of Twitter(Need to edit the Classifier.java to change the output from Tweet to Action in System.out.println in src/main/java/com/chimpler/example/bayes/Classifier.java Line no:112 )

****

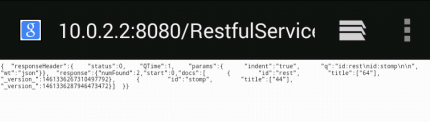
2) **Restful Webservice:**

**Running the restful web service in local host:**

Response for Stomps and Rest



**Running restful web service from android browser:**

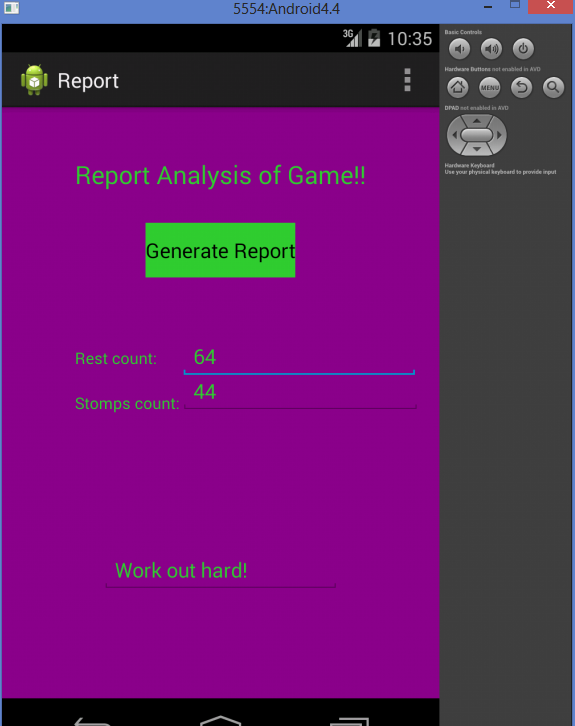
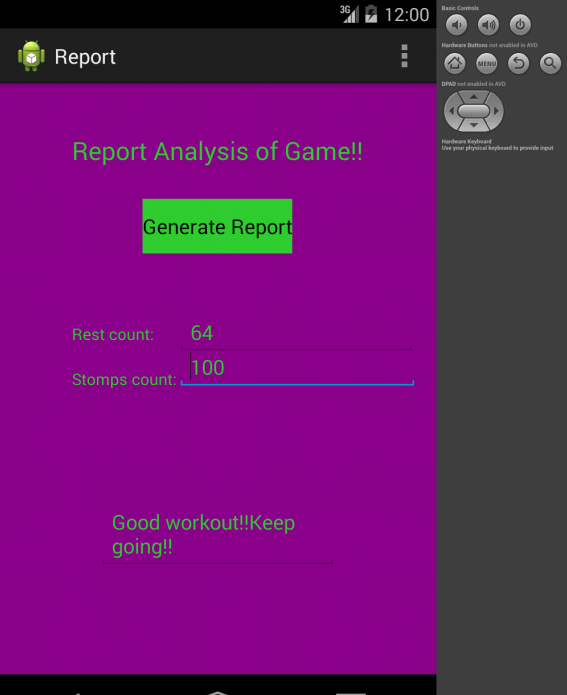


**Running restful webservice from android application:**

Retrieves the number of stomps and Rest from json and displays in Edit Text in android which would be more clear for the user instead of seeing in web view as a plain json data.

If the Stomps > Rest --------🡪 Good Job!!

If the Stomps < Rest ------------🡪 Work Hard!!

**Response JSON:**

{

"responseHeader":{

"status":0,

"QTime":0,

"params":{

"indent":"true",

"q":"id:rest\nid:stomp",

"wt":"json"}},

"response":{"numFound":2,"start":0,"docs":[

{

"id":"rest",

"title":["64"],

"\_version\_":1461336267310497792},

{

"id":"stomp",

"title":["100"],

"\_version\_":1461342652232368128}]

}

**3) Short Report:**

Played the sensor game to collect the axis(x,y,z) values. Detected the rest and stomp values based on the threshold of y axis values. Trained the training set with action, unique id and the y axis. (All axis values and timestamp can be included later for training). The sample **training dataset:**

rest 1 -1.02

rest 2 -1.03

stomp 3 1.67

stomp 4 1.94

**After sequencing:**

/rest/1 -1.02

/rest/2 -1.03

/stomp/3 1.67

/stomp/4 1.94

Commands:

To sequence the data with java file:

* java -cp mahout-naive-bayes-example-master/target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar com.chimpler.example.bayes.TweetTSVToSeq all\_data.txt trial8\_seq

Adding the sequenced file to hadoop

* hadoop fs -put trial8\_seq

Vectoring the sequenced data

* mahout seq2sparse -i trial8\_seq -o trail8\_vectors

splitting the data(40% of dataset for testing)

* mahout split -i trail8\_vectors/tfidf-vectors --trainingOutput trail8\_train\_vec --testOutput trail8\_test\_vec --randomSelectionPct 40 --overwrite --sequenceFiles -xm sequential

Training the data-set (mode,labelindex derived)

* mahout trainnb -i trail8\_train\_vec -el -li labelindextrail8 -o modeltrial8 -ow –c

Testing the dataset with the existing split data set

* mahout testnb -i trail8\_test\_vec -l labelindextrail8 -m modeltrial8 -ow -o trail8\_testing –c

Copying the model, labelindex and df-count to local:

* hadoop fs -get labelindextrail8 labelindex
* hadoop fs -get modeltrial8 model
* hadoop fs -get trail8\_vectors/dictionary.file-0 dictionary.file-0
* hadoop fs -getmerge trail8\_vectors/df-count df-count

Classifying the new set of data:

* java -cp target/twitter-naive-bayes-example-1.0-jar- with-dependencies.jar com.chimpler.example.bayes.Classifier model labelindex dictionary.file-0 df-count data\_classify > result.txt

Wrote a class to read the result.txt to json data and invoked the URL to upload the json to <http://134.193.136.127:8983/solr> , have a small error(working on it).

For now , added the result of classifier to Solr directly in <http://134.193.136.127:8983/solr> and created a Restful service to retrieve the number of stomps and number of Rest and displayed it in Android application.