5590RA Group PG2 Project Report #1:

By:
Bruno Vizcarra
Rishabh Bhojak
Gharib Gharibi
Sravan Mekarthi

Due: October 3rd, 2014 by midnight

VISUALIZING PUBLIC SENTIMENT OF THE IPHONE 6 AND IPHONE 6 PLUS

Introduction:

The basis for the first phase of this project is to create a software platform that will:

- Collect real-time Twitter information
- Filter this information to a usable format
- Analyze the filtered data using a machine learning algorithm
- Output the processed data as an easy to visualize format

For our specific project, a sentiment analysis of social data (tweets) was to be implemented to measure public opinion regarding the release of the new iPhone 6 and iPhone 6 Plus mobile devices manufactured by Apple Corporation.

1) Design:

- Data model - The overall architecture of our design can be viewed as follows:

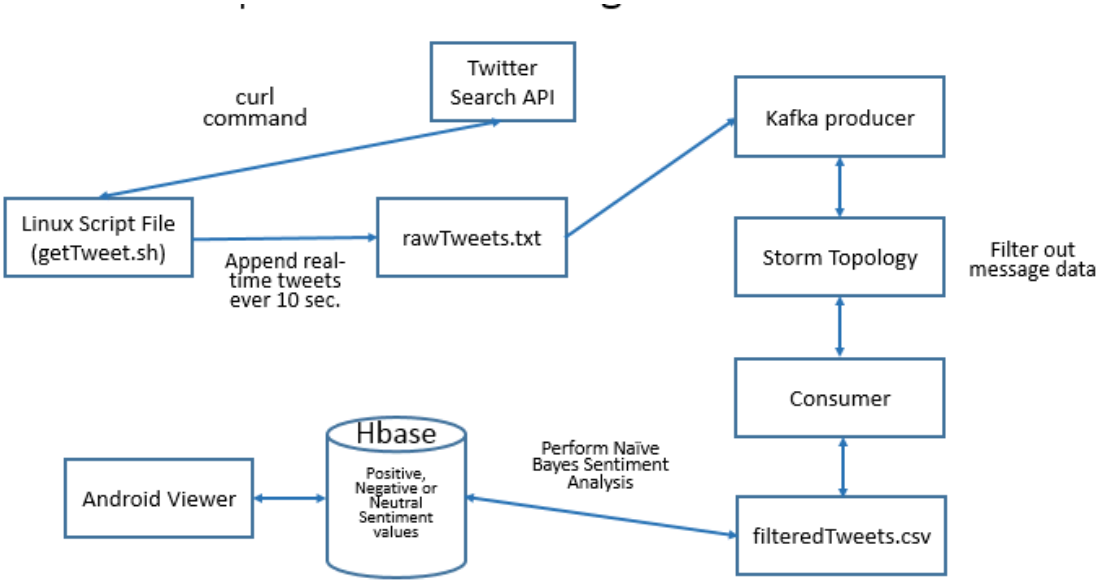

Figure 1: Real-time data flow from Twitter to Android

Using a simple bash script file, getTweet.sh, we are able to pull in information from the Twitter Search API using the curl function (shown below) specified in Open Authentication (OAuth) section of app.twitter.com. Here we see that "#iphone6" OR "#iphone6plus" are the terms being searched for and pushed to our data file:

```
curl --get 'https://api.twitter.com/1.1/search/tweets.json' --data
'f=realtime&lang=en&q=%23iphone6+OR+%23iphone6plus' --header 'Authorization: OAuth
oauth_consumer_key="7LvwJH1U0ix2wbLAmY9imNdkR",
oauth_nonce="3ccd0d06babdbc09013dc0a76be48502",
oauth_signature="rjqCLO%2FUNA125lFOrBXO8Z8KUbg%3D",
oauth_signature_method="HMAC-SHA1", oauth_timestamp="1411775117",
oauth_token="2811560934-t2ZfEBtpNlX9kmmUdTcFS9i4dm3Z2Dz20O7Y6Ln",
oauth_version="1.0"' --verbose >> rawTweets.txt
```

Figure 2 shows the output of this raw text file after this operation. The yellow highlights contain the pertinent data that we wish to filter out, the tweet text field. As shown in the very first highlight of Figure 2, we notice a negative sentiment regarding the iPhone 6.

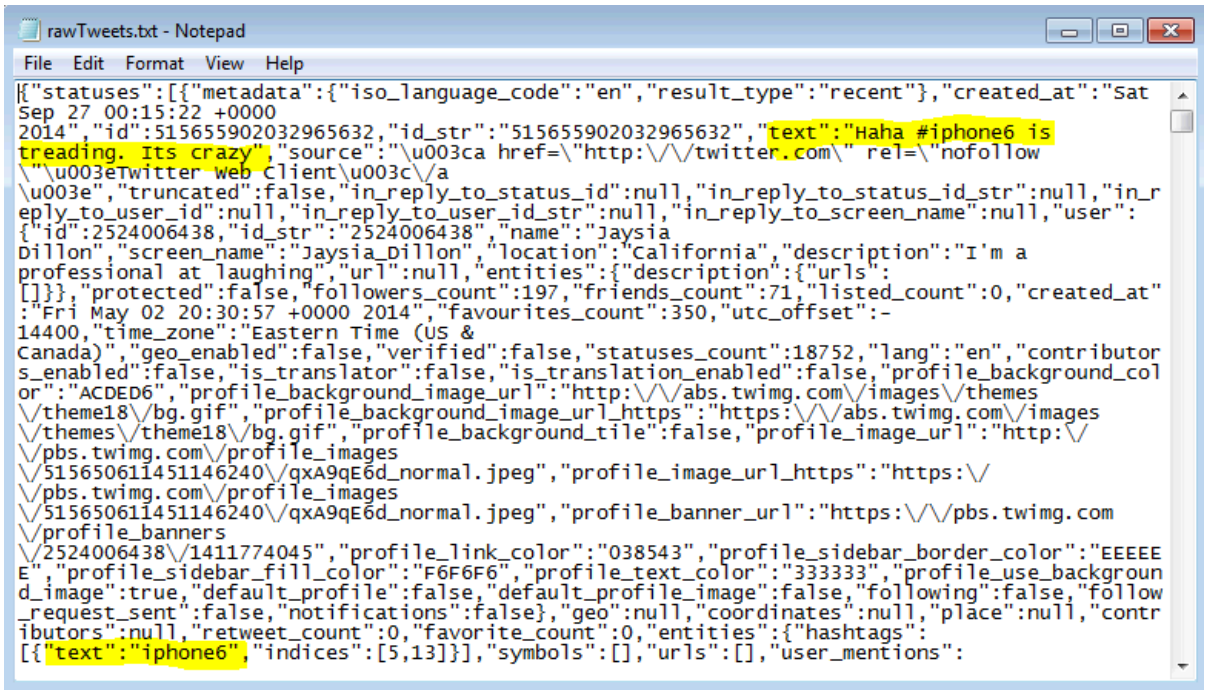

Figure 2: Raw text file containing real-time tweets

After collecting the tweets in the above text file. We had to filter them and purse the files into the JSon format that our Kafka producer can understand. SO we had to develop another java program that will read the above text file and extract the tweets with the corresponding “id” only. Because we are looking for the tweets regarding the users or any other metadata.

```
package com.demo;

import java.io.File;
import java.io.PrintWriter;
import java.util.Scanner;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
public class TweetParser {

	public static void main(String[] args) throws Exception{

		Scanner scan = new Scanner(new File("tweet.txt"));
		PrintWriter writer = new PrintWriter("output_tweet.txt", "UTF-
8");


		while(scan.hasNext()){
			String line = scan.nextLine();
			JSONParser parser = new JSONParser();
			try{
			JSONObject tweet = (JSONObject) parser.parse(line);

			if(tweet.containsKey("created_at")){
				String id = tweet.get("id").toString();
				String text = tweet.get("text").toString();
				JSONObject newTweet = new JSONObject();
				newTweet.put("id", id);
				newTweet.put("text", text);
				System.out.println(newTweet.toString());
				writer.println(newTweet.toString());


			}
			}catch(Exception e){
			e.printStackTrace();
			}

		}

		writer.close();


	}

}
```

We pass the rawTweets.txt file to our Twitter Storm topology where the data is filter and analyzed for sentiment then stored to an Hbase database. Figure 3 shows this topology and Figure 4 shows an alternate topology used.

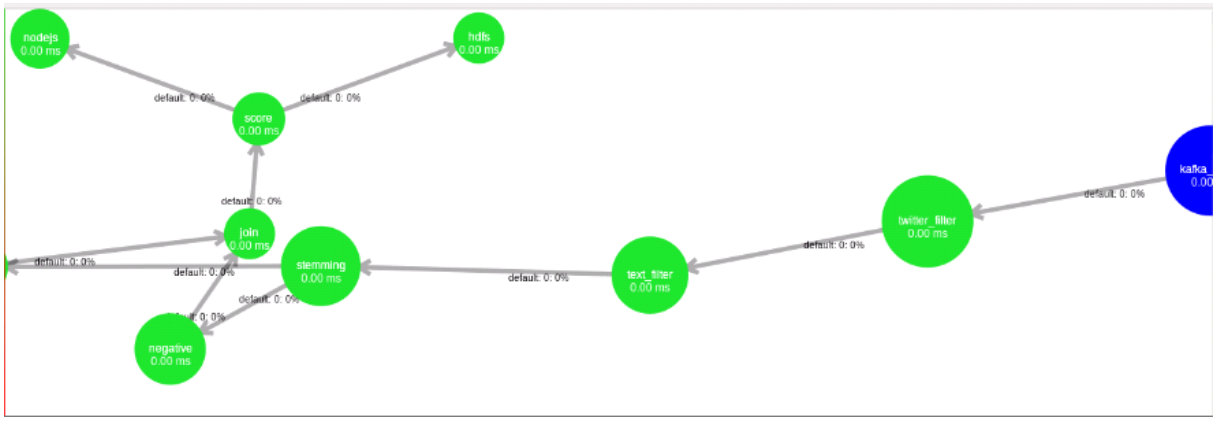

Figure 3: Twitter Storm Sentiment Complex Topology

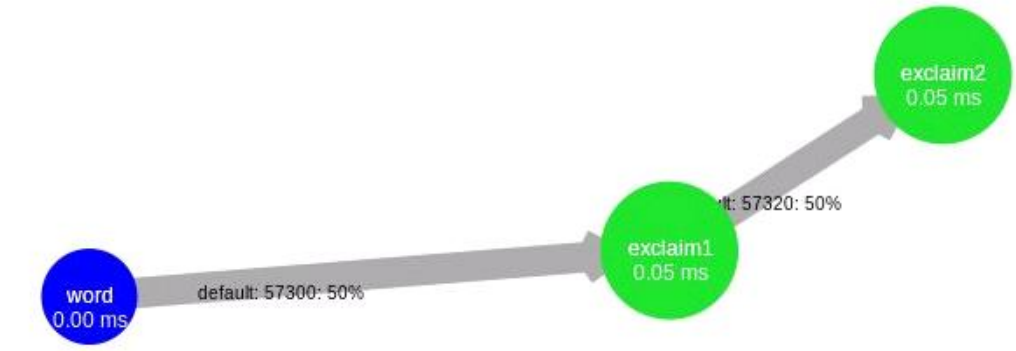

Figure 4: Twitter Storm Sentiment Simple Topology

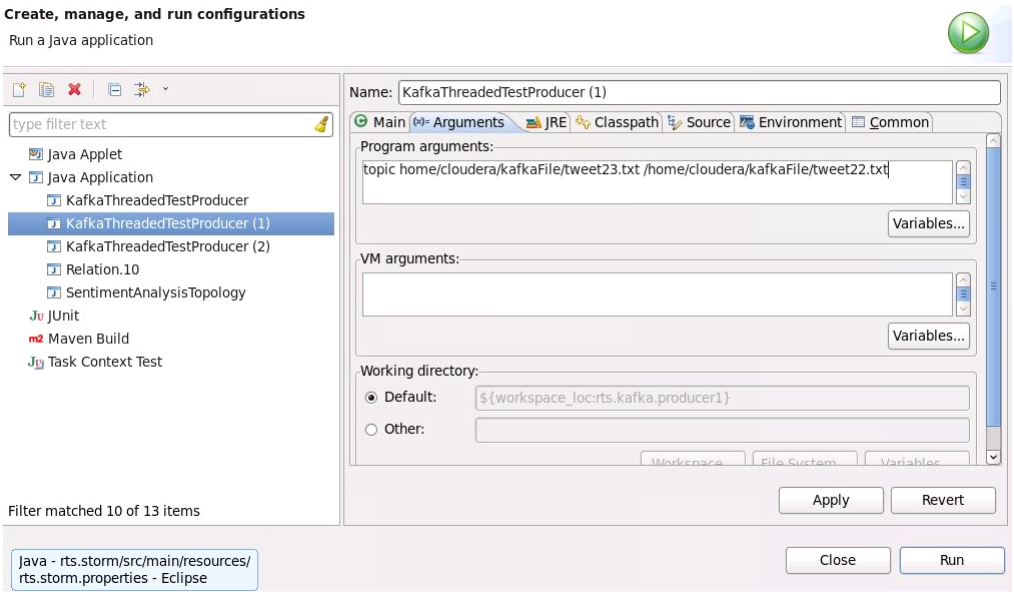

Figure 5: Kafka producer topic and input files

- Sentiment analysis model and algorithm – The Naïve Bayes algorithm is used for sentiment analysis.

- Predictive/recommendation model and algorithm – There were no predictive algorithms employed in this challenge.

- Selection of datasets – No other datasets other than those provided by Twitter were incorporated in this challenge.

- Mobile App/Web design – Both an Android application and web viewer were implemented to visualize our final output. Figure 6 shows the output file compiled after both the CSVTopology and MachineLearningTopology expressing the product overall sentiment.

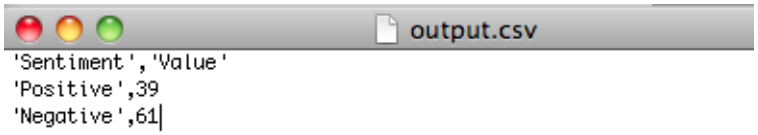

Figure 6: Output file displaying overall sentiment

This output file is written to a virtual directory on our Tomcat web server where it is read by our web viewer (Appendix - Source Code – index.html) using a Google Charts API and output to the screen as shown in Figure 7.

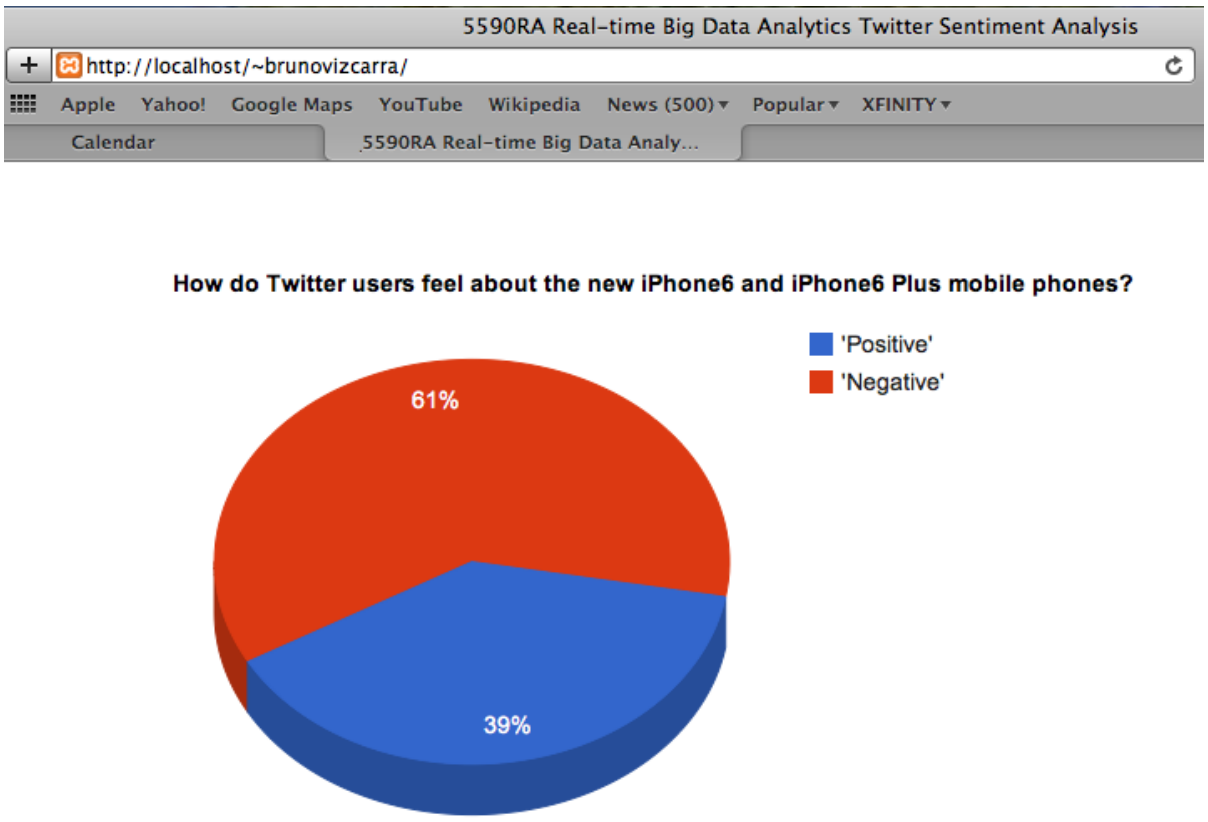

Figure 7: Web visualization using google charts API

2) **Features Implemented**
   - Sentiment analysis algorithms - The Naïve Bayes algorithm is used for sentiment analysis.
   - Mobile User Interface – No specific features were implemented into the user interface other than default settings for visualization as explained above.

3) **Outputs: description with screenshots of the features**
   No output was generated at this time.

4) **All the Web Service and Web Site URLs**
   Can be found on our own Cloudera server.

5) **Your midterm Github URL:**
   https://github.com/Gharibw/RTBD

6) **Limitations**
   Zookeeper could not be run on the class server for our group. This meant that Kafka producer and consumer could also not be implemented. Thus, all challenge codes had to be implemented on our own feeble Cloudera server.

7) **References**

[1] https://github.com/zdata-inc/StormSampleProject
[2] https://github.com/adarshms/sentweet
[3] https://developers.google.com/chart/interactive/docs/reference#addlistener
[4] http://stackoverflow.com/questions/14211636/how-to-use-google-chart-with-data-from-a-csv

8) **Source Code**

index.html (Visualizer)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta charset="utf-8" />
    <title>jQuery-CSV - Google Visualization Demo</title>
    <script src="http://code.jquery.com/jquery-1.8.2.js"></script>
    <script src="http://jquery-csv.googlecode.com/git/src/jquery.csv.js"></script>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

    // load the visualization library from Google and set a listener
    google.load("visualization", "1", {packages:["corechart"]});
    google.setOnLoadCallback(drawChart);

    // this has to be a global function
    function drawChart() {
      // grab the CSV
      $.get("output.csv", function(csvString) {
        // transform the CSV string into a 2-dimensional array
        var arrayData = $.csv.toArrays(csvString, {onParseValue: $.csv.hooks.castToScalar});

        // this new DataTable object holds all the data
        var data = new google.visualization.arrayToDataTable(arrayData);

        // this view can select a subset of the data at a time
        var view = new google.visualization.DataView(data);
        view.setColumns([0,1]);

       // set chart options
       var options = {
         title: "How do Twitter users feel about the new iPhone6 and iPhone6 Plus mobile phones?",
         hAxis: {title: data.getColumnLabel(0), minValue: data.getColumnRange(0).min, maxValue:
data.getColumnRange(0).max},
         vAxis: {title: data.getColumnLabel(1), minValue: data.getColumnRange(1).min, maxValue:
data.getColumnRange(1).max},
         legend: 'none',
         is3D: true,
         pieStartAngle: 100
       };

       // create the chart object and draw it
       var chart = new google.visualization.PieChart(document.getElementById('piechart'));
       chart.draw(view, options);
```

```
    });
   }
   </script>
  </head>
  <body>
   <div id="piechart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Retriv.java

```
package t_data;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import twitter4j.FilterQuery;

import twitter4j.*;
import twitter4j.conf.ConfigurationBuilder;
public class Retriv {

/**
* @param args
*/
private static ConfigurationBuilder cb;
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Twitter Streaming");
cb = new ConfigurationBuilder();
cb.setDebugEnabled(true);
cb.setOAuthConsumerKey("mzUbFQwHZijQDcShr9VTywhyC");
cb.setOAuthConsumerSecret("ZYWEiuQC1NYtyNEhVzVWNckLdpNZwv54iFUN7dgTxxERIbOwU
B");
cb.setOAuthAccessToken("2310210140-XeAC5XdAUyxnMrd2wkPQ1YtCVMTYOvdGv03pvSv");
cb.setOAuthAccessTokenSecret("Xo9UdAXsIX2aG2a7Xfze2mf9QHMuQbQoWdvwnuHFEMyRl");
TwitterStream t_new= new TwitterStreamFactory(cb.build()).getInstance();

StatusListener s_new= new StatusListener(){
@Override
public void onException(Exception ex) {
// TODO Auto-generated method stub
ex.printStackTrace();
}

@Override
public void onDeletionNotice(StatusDeletionNotice sdn) {
// TODO Auto-generated method stub
System.out.println("Got a status deletion notice id:" + sdn.getStatusId());
```

```
}

@Override
public void onScrubGeo(long userId,long upToStatusId) {
// TODO Auto-generated method stub
System.out.println("Got scrub_geo event userId:" + userId + " upToStatusId:" + upToStatusId);
}

@Override
public void onStallWarning(StallWarning arg0) {
// TODO Auto-generated method stub
}

@Override
public void onStatus(Status status){
// TODO Auto-generated method stub
if(!status.getLang().contains("en") && !status.getText().contains("#iPhone6")){

            return;
        }
System.out.println("@" + status.getUser().getScreenName() + " - " + status.getText());

String s=status.getLang()+"{\"id\":"+status.getId()+", \"text\": "+status.getText()+"},"+"\n";
fwrite(s);
}
@Override
public void onTrackLimitationNotice(int ns) {
// TODO Auto-generated method stub
System.out.println("Got track limitation notice:" + ns);
}
public void fwrite(String msg)
{
FileWriter f;
try {
f = new FileWriter("Input.txt",true);
BufferedWriter bw= new BufferedWriter(f);
bw.write(msg);
bw.close();
} catch (IOException e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
}
}
};
t_new.addListener(s_new);
t_new.filter(new FilterQuery().language(new String[]{"en"}));
t_new.sample();
}
```