**Source Code**

Name: L V Sandeep Bathina(3)

Rakesh Vistarakula (16)

L Dheeraj Polisetty (13)

Balaji Gannavarapu (7)

1. **Find\_all\_scriptures.py**

# autogenerated by regex.py

import re

find\_all\_scriptures = re.compile("""

(

(?P<halloween>hal\w{0,6}\.?) # halloween

|(?P<expandables>exp\w{0,8}\.?) # expandables

|(?P<lucy>luc\w{0,1}\.?) # lucy

|(?P<annabelle>ann\w{0,6}\.?) # annabelle

|(?P<nemo>nem\w{0,1}\.?) # nemo

|(?P<titanic>tit\w{0,4}\.?) # titanic

|(?P<aagadu>aag\w{0,3}\.?) # aagadu

|(?P<gav>gav\w{0,0}\.?) # gav

|(?P<i>I\w{0,0}\.?) # i

|(?P<avengers>ave\w{0,5}\.?) # Avengers

)

\s+(?P<verse>\d{1,3}\s\*:\s\*\d{1,3})

""",re.VERBOSE|re.MULTILINE).finditer

def filtergroupdict(ma):

"""Remove all entries with values == None"""

di = filter(lambda x: x[1]!=None and x[0]!='verse', ma.groupdict().items())

if len(di) != 1:

return None

ret = {}

ret['book'] = di[0][0]

ret['verse'] = ma.groupdict()['verse']

ret['verse'] = ret['verse'].replace(' ','')

return ret

1. **Printer Bolt.java**

package technicalelvis.habakkuk.bolt;

import backtype.storm.topology.BasicOutputCollector;

import backtype.storm.topology.OutputFieldsDeclarer;

import backtype.storm.topology.base.BaseBasicBolt;

import backtype.storm.tuple.Tuple;

import java.io.\*;

public class PrinterBolt extends BaseBasicBolt {

File file = new File("/home/group8/Twitter.txt");

File file2 = new File("/home/group8/Image.txt");

FileWriter fw = null;

BufferedWriter bw = null;

FileWriter fw2 = null;

BufferedWriter bw2 = null;

long id;

String text = null;

@Override

public void execute(Tuple tuple, BasicOutputCollector collector) {

try{

if(!file.exists())

{

file.createNewFile();

}

fw = new FileWriter(file.getAbsoluteFile(),true);

bw = new BufferedWriter(fw);

bw.write("\n"+tuple);

bw.close();

}catch(IOException e){

e.printStackTrace();

}

}

@Override

public void declareOutputFields(OutputFieldsDeclarer ofd) {

}

}

1. **TwitterSampleSpout.java**

package technicalelvis.habakkuk.spout;

import java.text.SimpleDateFormat;

import java.util.Map;

import java.util.Properties;

import java.util.concurrent.LinkedBlockingQueue;

import java.util.HashMap;

import org.apache.log4j.Logger;

import com.fasterxml.jackson.databind.ObjectMapper;

import twitter4j.FilterQuery;

import twitter4j.MediaEntity;

import twitter4j.Status;

import twitter4j.StatusDeletionNotice;

import twitter4j.StatusListener;

import twitter4j.TwitterStream;

import twitter4j.TwitterStreamFactory;

import twitter4j.json.DataObjectFactory;

import twitter4j.auth.AccessToken;

import twitter4j.conf.ConfigurationBuilder;

import twitter4j.StallWarning;

import backtype.storm.Config;

import backtype.storm.spout.SpoutOutputCollector;

import backtype.storm.task.TopologyContext;

import backtype.storm.topology.OutputFieldsDeclarer;

import backtype.storm.topology.base.BaseRichSpout;

import backtype.storm.tuple.Fields;

import backtype.storm.tuple.Values;

import backtype.storm.utils.Utils;

// TODO: twitter4j with API keys

public class TwitterSampleSpout extends BaseRichSpout {

// static Logger LOG = Logger.getLogger(TwitterSampleSpout.class);

SpoutOutputCollector \_collector;

LinkedBlockingQueue<Status> queue = null;

TwitterStream \_twitterStream;

String \_username;

String \_pwd;

String consumer\_key;

String consumer\_secret;

String access\_token;

String token\_secret;

/\* Used for Oauth \*/

public TwitterSampleSpout(Properties props) {

consumer\_key = props.getProperty("twitter4j.consumer\_key");

consumer\_secret = props.getProperty("twitter4j.consumer\_secret");

access\_token = props.getProperty("twitter4j.access\_token");

token\_secret = props.getProperty("twitter4j.token\_secret");

}

@Override

public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {

queue = new LinkedBlockingQueue<Status>(1000);

\_collector = collector;

StatusListener listener = new StatusListener() {

@Override

public void onStatus(Status status) {

queue.offer(status);

}

@Override

public void onDeletionNotice(StatusDeletionNotice sdn) {

}

@Override

public void onTrackLimitationNotice(int i) {

}

@Override

public void onScrubGeo(long l, long l1) {

}

@Override

public void onException(Exception e) {

}

@Override

public void onStallWarning(StallWarning warning){

}

};

ConfigurationBuilder cb = new ConfigurationBuilder();

TwitterStreamFactory fact = new TwitterStreamFactory();

\_twitterStream = fact.getInstance();

\_twitterStream.setOAuthConsumer(consumer\_key, consumer\_secret);

\_twitterStream.setOAuthAccessToken(new AccessToken(access\_token, token\_secret));

\_twitterStream.addListener(listener);

//Filter

FilterQuery filtre = new FilterQuery();

String[] keywordsArray = {"Annabelle", "movie"};

filtre.track(keywordsArray);

\_twitterStream.filter(filtre);

//!\_twitterStream.sample();

}

@Override

public void nextTuple() {

Status ret = queue.poll();

if(ret==null) {

Utils.sleep(50);

} else {

/\* Map<String, String> data = new HashMap<String, String>();

// user info

data.put("username", ret.getUser().getName());

data.put("screenname", ret.getUser().getScreenName());

data.put("follower\_count", Integer.toString(ret.getUser().getFollowersCount()));

data.put("friends\_count", Integer.toString(ret.getUser().getFriendsCount()));

data.put("location",ret.getUser().getLocation());

data.put("lang", ret.getUser().getLang());

data.put("user\_url", ret.getUser().getURL());

// tweet info

data.put("text", ret.getText());

data.put("tweetid",Long.toString(ret.getId()));

data.put("retweet\_count",Long.toString(ret.getRetweetCount()));

// format created\_at as YYYY-MM-dd

data.put("created\_at", ret.getCreatedAt().toString());

SimpleDateFormat \_format = new SimpleDateFormat("yyyy-MM-dd");

StringBuilder \_datestr = new StringBuilder(\_format.format(ret.getCreatedAt()));

data.put("created\_at\_date", \_datestr.toString());

data.put("created\_at", Long.toString(ret.getCreatedAt().getTime()));

// emit tuple to next bolt

\_collector.emit(new Values(data));\*/

for(MediaEntity mediaEntity:ret.getMediaEntities())

{

\_collector.emit(new Values(mediaEntity.getMediaURL())) ;

}

}

}

@Override

public void close() {

\_twitterStream.shutdown();

}

@Override

public Map<String, Object> getComponentConfiguration() {

Config ret = new Config();

ret.setMaxTaskParallelism(1);

return ret;

}

@Override

public void ack(Object id) {

}

@Override

public void fail(Object id) {

}

@Override

public void declareOutputFields(OutputFieldsDeclarer declarer) {

declarer.declare(new Fields("tweet"));

}

**}**

1. **Weka.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import java.util.TreeMap;

import weka.classifiers.Classifier;

import weka.classifiers.bayes.NaiveBayes;

import weka.core.Attribute;

import weka.core.Instance;

import weka.core.Instances;

import weka.core.SparseInstance;

import weka.core.converters.CSVLoader;

import cmu.arktweetnlp.POSTagger;

import cmu.arktweetnlp.Token;

import com.csvreader.CsvReader;

/\*\*

\*

\* @author girish

\*/

public class WekaTutorial {

private ArrayList<String> featureWords;

private ArrayList<Attribute> attributeList;

private Instances inputDataset;

private POSTagger posTagger;

private Classifier classifier;

private ArrayList<String> sentimentClassList;

public WekaTutorial()

{

attributeList = new ArrayList<>();

posTagger = new POSTagger();

initialize();

}

private void initialize()

{

ObjectInputStream ois = null;

try {

//reads the feature words list to a hashset

ois = new ObjectInputStream(new FileInputStream("FeatureWordsList.dat"));

featureWords = (ArrayList<String>) ois.readObject();

} catch (Exception ex) {

System.out.println("Exception in Deserialization");

} finally {

try {

ois.close();

} catch (IOException ex) {

System.out.println("Exception while closing file after Deserialization");

}

}

//creating an attribute list from the list of feature words

sentimentClassList = new ArrayList<>();

sentimentClassList.add("positive");

sentimentClassList.add("negative");

for(String featureWord : featureWords)

{

attributeList.add(new Attribute(featureWord));

}

//the last attribute reprsents ths CLASS (Sentiment) of the tweet

attributeList.add(new Attribute("Sentiment",sentimentClassList));

}

public void trainClassifier(final String INPUT\_FILENAME)

{

getTrainingDataset(INPUT\_FILENAME);

//trainingInstances consists of feature vector of every input

Instances trainingInstances = createInstances("TRAINING\_INSTANCES");

for(Instance currentInstance : inputDataset)

{

//extractFeature method returns the feature vector for the current input

Instance currentFeatureVector = extractFeature(currentInstance);

//Make the currentFeatureVector to be added to the trainingInstances

currentFeatureVector.setDataset(trainingInstances);

trainingInstances.add(currentFeatureVector);

}

//You can create the classifier that you want. In this tutorial we use NaiveBayes Classifier

//For instance classifier = new SMO;

classifier = new NaiveBayes();

try {

//classifier training code

classifier.buildClassifier(trainingInstances);

//storing the trained classifier to a file for future use

weka.core.SerializationHelper.write("NaiveBayes.model",classifier);

} catch (Exception ex) {

System.out.println("Exception in training the classifier.");

}

}

public void testClassifier(final String INPUT\_FILENAME)

{

getTrainingDataset(INPUT\_FILENAME);

//trainingInstances consists of feature vector of every input

Instances testingInstances = createInstances("TESTING\_INSTANCES");

for(Instance currentInstance : inputDataset)

{

//extractFeature method returns the feature vector for the current input

Instance currentFeatureVector = extractFeature(currentInstance);

//Make the currentFeatureVector to be added to the trainingInstances

currentFeatureVector.setDataset(testingInstances);

testingInstances.add(currentFeatureVector);

}

try {

//Classifier deserialization

classifier = (Classifier) weka.core.SerializationHelper.read("NaiveBayes.model");

try {

//String content = "This is the content to write into file";

File file = new File("/home/cloudera/Desktop/androidtemp.csv");

// if file doesnt exists, then create it

if (!file.exists()) {

file.createNewFile();

}

FileWriter fw = new FileWriter(file.getAbsoluteFile(),true);

BufferedWriter bw = new BufferedWriter(fw);

// String content = testingInstances.attribute("Sentiment").value((int)score).toString();

bw.write("sentiment");

bw.write("\n");

bw.close();

System.out.println("Done");

}catch (IOException e) {

e.printStackTrace();

}

//classifier testing code

for(Instance testInstance : testingInstances)

{

double score = classifier.classifyInstance(testInstance);

// System.out.println(testingInstances.attribute("Sentiment").value((int)score));

/\* try {

CsvReader products = new CsvReader("testing.csv");

products.readHeaders();

while (products.readRecord())

{

String productID = products.get("Twitter");

products.get(0);

// perform program logic here

System.out.println(products.get(0)+","+testingInstances.attribute("Sentiment").value((int)score));

}

products.close();

} catch (IOException e) {

e.printStackTrace();

}\*/

try {

//String content = "This is the content to write into file";

File file = new File("/home/cloudera/Desktop/androidtemp.csv");

// if file doesnt exists, then create it

if (!file.exists()) {

file.createNewFile();

}

FileWriter fw = new FileWriter(file.getAbsoluteFile(),true);

BufferedWriter bw = new BufferedWriter(fw);

String content = testingInstances.attribute("Sentiment").value((int)score).toString();

System.out.println(testingInstances.attribute("Sentiment").value((int)score).toString());

bw.write(content);

bw.write("\n");

bw.close();

}catch (IOException e) {

e.printStackTrace();

}

}

System.out.println("Done");

}

catch (Exception ex) {

System.out.println("Exception in testing the classifier.");

}

}

private void getTrainingDataset(final String INPUT\_FILENAME)

{

try{

//reading the training dataset from CSV file

CSVLoader trainingLoader =new CSVLoader();

trainingLoader.setSource(new File(INPUT\_FILENAME));

inputDataset = trainingLoader.getDataSet();

}catch(IOException ex)

{

System.out.println("Exception in getTrainingDataset Method");

}

}

private Instances createInstances(final String INSTANCES\_NAME)

{

//create an Instances object with initial capacity as zero

Instances instances = new Instances(INSTANCES\_NAME,attributeList,0);

//sets the class index as the last attribute (positive or negative)

instances.setClassIndex(instances.numAttributes()-1);

return instances;

}

private Instance extractFeature(Instance inputInstance)

{

Map<Integer,Double> featureMap = new TreeMap<>();

List<Token> tokens = posTagger.runPOSTagger(inputInstance.stringValue(0));

for(Token token : tokens)

{

switch(token.getPOS())

{

case "A":

case "V":

case "R":

case "#":

// System.out.println(token.getWord());

String word = token.getWord().replaceAll("#","");

if(featureWords.contains(word))

{

//adding 1.0 to the featureMap represents that the feature word is present in the input data

featureMap.put(featureWords.indexOf(word),1.0);

}

}

}

int indices[] = new int[featureMap.size()+1];

double values[] = new double[featureMap.size()+1];

int i=0;

for(Map.Entry<Integer,Double> entry : featureMap.entrySet())

{

indices[i] = entry.getKey();

values[i] = entry.getValue();

i++;

}

indices[i] = featureWords.size();

values[i] = (double)sentimentClassList.indexOf(inputInstance.stringValue(1));

return new SparseInstance(1.0,values,indices,featureWords.size());

}

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

{

WekaTutorial wekaTutorial = new WekaTutorial();

wekaTutorial.trainClassifier("training.csv");

wekaTutorial.testClassifier("testing.csv");

try {

CsvReader products = new CsvReader("testing.csv");

CsvReader products1 = new CsvReader("/home/cloudera/Desktop/androidtemp.csv");

products.readHeaders();

products1.readHeaders();

while (products.readRecord() && products1.readRecord())

{

String productID = products.get("Twitter");

products.get(0);

String productID1 = products1.get("sentiment");

// perform program logic here

String output = products.get(0)+"/"+products1.get(0);

try {

//String content = "This is the content to write into file";

File file = new File("/home/cloudera/Desktop/androidop.txt");

// if file doesnt exists, then create it

if (!file.exists()) {

file.createNewFile();

}

FileWriter fw = new FileWriter(file.getAbsoluteFile(),true);

BufferedWriter bw = new BufferedWriter(fw);

bw.write(output);

bw.write("\n");

bw.close();

}catch (IOException e) {

e.printStackTrace();

}

}

products.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

1. **Androidapp**

**package** com.example.androidapp;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileReader;

**import** java.util.StringTokenizer;

**import** android.os.Bundle;

**import** android.os.Environment;

**import** android.support.v7.app.ActionBarActivity;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

**import** android.widget.TextView;

**public** **class** MainActivity **extends** ActionBarActivity {

**private** String twitterda;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

Button button1 =(Button) findViewById(R.id.*button1*);

button1.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

FileInputStream file;

**try**{

File fa = **new** File(Environment.*getExternalStorageDirectory*()+"/Data1/Read.txt");

**if**(fa.exists()==**true**){

file = **new** FileInputStream(fa);

BufferedReader bufferr = **new** BufferedReader(**new** FileReader(fa));

String text;

String da=**new** String();

String st = "";

**while**((text = bufferr.readLine())!=**null**)

{

st = text;

da=da.concat(st+"\n");

}

// System.out.println(data);

//read data from the file and store

twitterda=da;

String result=Analysys(da);

TextView textview=(TextView)findViewById(R.id.*textView1*);

textview.setText(da);

file.close();

bufferr.close();

}

}

**catch**(Exception e){

System.*out*.println("exception in reading file");

}

}

});

Button button2 =(Button) findViewById(R.id.*button2*);

button2.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

String result=Analysys(twitterda);

TextView textview1=(TextView)findViewById(R.id.*textView1*);

textview1.setText(result);

}

});

}

**public** String Analysys(String data)

{

**int** count1=0;

**int** count2=0;

**try** {

StringTokenizer st = **new** StringTokenizer(data,"/");

**while** (st.hasMoreTokens()) {

String cou=st.nextToken();

**if**(cou.contains("positive")){

count1++;

}

**else** **if**(cou.contains("negative")){

count2++;

}

}

/\*End of while\*/

String analysys=**new** String("Positive Tweets : "+count1+"\t"+" Negative tweets : "+count2);

data=analysys;

Intent bargraph = new Intent(Report.this,Bargraph.class);

System.out.println("Bargraph");

startActivity(bargraph);

} **catch** (Exception e) {

e.printStackTrace();

}

**return** data;

}

}

**Piechart.java**

package com.example.androidapp;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import org.achartengine.ChartFactory;

import org.achartengine.model.CategorySeries;

import org.achartengine.renderer.DefaultRenderer;

import org.achartengine.renderer.SimpleSeriesRenderer;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Environment;

import android.view.Menu;

public class Bargraph extends Activity {

public String[] mMonth = new String[] {

"Jan", "Feb" , "Mar", "Apr", "May", "Jun",

"Jul", "Aug" , "Sep", "Oct", "Nov", "Dec"

};

String day;

Context context=this;

public String line;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.bargraph);

openChart();

openChart1();

// Getting reference to the button btn\_chart

/\* Button btnChart = (Button) findViewById(R.id.btn\_chart);

// Defining click event listener for the button btn\_chart

OnClickListener clickListener = new OnClickListener() {

@Override

public void onClick(View v) {

// Draw the Income vs Expense Chart

openChart();

}

};\*/

// Setting event click listener for the button btn\_chart of the MainActivity layout

//btnChart.setOnClickListener(clickListener);

}

public void openChart(){

// Pie Chart Section Names

String[] code = new String[] {

"Left", "Right", "Up", "Down"

};

//System.out.println("IN BARGRAPH");

try{

File f = new File(Environment.getExternalStorageDirectory()+"/Data2/sensor.txt");

//FileInputStream fis = new FileInputStream(f);

BufferedReader bufferr = new BufferedReader(new FileReader(f));

System.out.println(f.exists());

String sp = "";

while((line = bufferr.readLine())!=null)

{

sp = line;

System.out.println(sp);

String[] splits = sp.split("\t");

day=Report.getDefaults("day", context);

System.out.println("day is currently"+day+splits[0]);

if(splits[0].equalsIgnoreCase(day))

{

System.out.println(splits[0]);

// System.out.println(Double.parseDouble(splits[0]));

Double[] distribution = {Double.parseDouble(splits[1]),Double.parseDouble(splits[2]),Double.parseDouble(splits[3]),Double.parseDouble(splits[4])};

// Pie Chart Section Value

//double[] distribution = { 3.9, 12.9, 55.8, 1.9, 23.7, 1.8 } ;

// Color of each Pie Chart Sections

int[] colors = { Color.BLUE, Color.MAGENTA, Color.GREEN, Color.CYAN, Color.RED,

Color.YELLOW };

// Instantiating CategorySeries to plot Pie Chart

CategorySeries distributionSeries = new CategorySeries(" Pie Chart Distribution");

System.out.println(distribution.length);

for(int i=0 ;i < distribution.length;i++){

// Adding a slice with its values and name to the Pie Chart

distributionSeries.add(code[i], distribution[i]);

}

// Instantiating a renderer for the Pie Chart

DefaultRenderer defaultRenderer = new DefaultRenderer();

for(int i = 0 ;i<distribution.length;i++){

SimpleSeriesRenderer seriesRenderer = new SimpleSeriesRenderer();

seriesRenderer.setColor(colors[i]);

seriesRenderer.setDisplayChartValues(true);

// Adding a renderer for a slice

defaultRenderer.addSeriesRenderer(seriesRenderer);

}

String t = "Pie Chart Analysis as on "+splits[0];

defaultRenderer.setChartTitle(t);

defaultRenderer.setChartTitleTextSize(20);

defaultRenderer.setZoomButtonsVisible(true);

// Creating an intent to plot bar chart using dataset and multipleRenderer

Intent intent = ChartFactory.getPieChartIntent(getBaseContext(), distributionSeries , defaultRenderer, "AChartEnginePieChartDemo");

//System.out.println("CHECKKKK");

// Start Activity

startActivity(intent);

break;

}

}

}

catch(Exception e){}

}

public void openChart1(){

// Pie Chart Section Names

String[] code = new String[] {

"Win", "Loss"

};

//System.out.println("IN BARGRAPH");

try{

File f = new File(Environment.getExternalStorageDirectory()+"/Data2/sensor.txt");

//FileInputStream fis = new FileInputStream(f);

BufferedReader bufferr = new BufferedReader(new FileReader(f));

System.out.println(f.exists());

String sp = "";

while((line = bufferr.readLine())!=null)

{

sp = line;

System.out.println(sp);

String[] splits = sp.split("\t");

day=Report.getDefaults("day", context);

System.out.println("day is currently"+day+splits[0]);

if(splits[0].equalsIgnoreCase(day))

{

System.out.println(splits[0]);

// System.out.println(Double.parseDouble(splits[0]));

Double[] distribution = {Double.parseDouble(splits[1]),Double.parseDouble(splits[2])};

// Pie Chart Section Value

//double[] distribution = { 3.9, 12.9, 55.8, 1.9, 23.7, 1.8 } ;

// Color of each Pie Chart Sections

int[] colors = { Color.BLACK, Color.LTGRAY };

// Instantiating CategorySeries to plot Pie Chart

CategorySeries distributionSeries = new CategorySeries(" Pie Chart Distribution");

System.out.println(distribution.length);

for(int i=0 ;i < distribution.length;i++){

// Adding a slice with its values and name to the Pie Chart

distributionSeries.add(code[i], distribution[i]);

}

// Instantiating a renderer for the Pie Chart

DefaultRenderer defaultRenderer = new DefaultRenderer();

for(int i = 0 ;i<distribution.length;i++){

SimpleSeriesRenderer seriesRenderer = new SimpleSeriesRenderer();

seriesRenderer.setColor(colors[i]);

seriesRenderer.setDisplayChartValues(true);

// Adding a renderer for a slice

defaultRenderer.addSeriesRenderer(seriesRenderer);

}

String t = "Pie Chart Analysis as on "+splits[0];

defaultRenderer.setChartTitle(t);

defaultRenderer.setChartTitleTextSize(20);

defaultRenderer.setZoomButtonsVisible(true);

// Creating an intent to plot bar chart using dataset and multipleRenderer

Intent intent = ChartFactory.getPieChartIntent(getBaseContext(), distributionSeries , defaultRenderer, "AChartEnginePieChartDemo");

//System.out.println("CHECKKKK");

// Start Activity

startActivity(intent);

break;

}

}

}

catch(Exception e){}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

}

**URLS:**

<http://blog.csdn.net/hljlzc2007/article/details/13275441>

<http://gramatica.usc.es/~gamallo/artigos-web/TASS2013.pdf>

<http://www.csvreader.com/java_csv_samples.php>

<http://stackoverflow.com/questions/2346713/csv-row-reader-question>

<https://blackboard.umkc.edu/webapps/blackboard/content/listContent.jsp?course_id=_114034_1&content_id=_1371866_1&mode=reset>

<http://www.java2s.com/Code/Jar/j/Downloadjavacsvjar.htm>

**References:**

1. Printerbolt.java is same as challenge 1 code
2. Piechart visualization is also same as in challenge 1
3. Weka.java some part of code is changed in challenge 2