

CS5542 Big Data Apps and Analytics

LAB ASSIGNMENT #5 & #6

1. Spark and Smartphone/Watch Application

- Implement a smart application with big data analytics related to your project showing the collaboration between Spark and Smart Apps.
- Implement Twitter Streaming and perform word count on it and publish the results and showcase it in your Smart Phone/Watch Application.

Solution:

Gathered tweets from twitter related to disease and performed a word count program on them and displayed via an android application on a smart phone.



Figure 1. Screen shot

2. Spark ML Lib Application Perform a machine learning algorithm with the Twitter Streaming data to categorize each Tweet

a) Training datasets:

Collect different categories of Tweets related to your project.
(Categories can be based on HashTags /Subjects etc.)

b) Test data: the upcoming twitter stream

Solution:

Collected tweets from twitter using following line of code:

```
package com.databricks.apps.twitter_classifier

import java.io.File

import com.google.gson.Gson
import org.apache.spark.streaming.twitter.TwitterUtils
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}

/**
 * Collect at least the specified number of tweets into json text files.
 */
object Collect {
  private var numTweetsCollected = 0L
  private var partNum = 0
  private var gson = new Gson()

  def main(args: Array[String]) {
    // Process program arguments and set properties
    if (args.length < 3) {
      System.err.println("Usage: " + this.getClass.getSimpleName +
        "<outputDirectory> <numTweetsToCollect> <intervalInSeconds> <partitionsEachInterval>")
      System.exit(1)
    }
    val Array(outputDirectory, Utils.IntParam(numTweetsToCollect), Utils.IntParam(intervalSecs), Utils.IntParam(partitionsEachInterval)) =
      Utils.parseCommandLineWithTwitterCredentials(args)
    val outputDir = new File(outputDirectory.toString)
    if (outputDir.exists()) {
      System.err.println("ERROR - %s already exists: delete or specify another directory".format(
        outputDirectory))
      System.exit(1)
    }
    outputDir.mkdirs()

    println("Initializing Streaming Spark Context...")
    val conf = new SparkConf().setAppName(this.getClass.getSimpleName)
    val sc = new SparkContext(conf)
    val ssc = new StreamingContext(sc, Seconds(intervalSecs))

    val tweetStream = TwitterUtils.createStream(ssc, Utils.getAuth)
      .map(gson.toJson(_))

    tweetStream.foreachRDD((rdd, time) => {
      val count = rdd.count()
      if (count > 0) {
        val outputRDD = rdd.repartition(partitionsEachInterval)
        outputRDD.saveAsTextFile(outputDirectory + "/tweets_" + time.milliseconds.toString)
        numTweetsCollected += count
        if (numTweetsCollected > numTweetsToCollect) {
          System.exit(0)
        }
      }
    })

    ssc.start()
    ssc.awaitTermination()
  }
}
```

Figure 2. Tweet collection code

```

1
2
3
4 from collections import Counter
5
6 def category_histogram(texts, short_texts):
7     # Classify the bios and tweets with MonkeyLearn's topic classifier.
8     topics = classify_batch(texts, MONKEYLEARN_TOPIC_CLASSIFIER_ID)
9     # The histogram will keep the counters of how many texts fall in
10    # a given category.
11    histogram = Counter()
12    samples = {}
13
14    for classification, text, short_text in zip(topics, texts, short_texts):
15        # Join the parent and child category names in one string.
16        category = classification[0]['label'] + '/' + classification[1]['label']
17        probability = (classification[0]['probability'] *
18                      classification[1]['probability'])
19        MIN_PROB = 0.3
20        # Discard texts with a predicted topic with probability lower than a threshold
21        if probability < MIN_PROB:
22            continue
23        # Increment the category counter.
24        histogram[category] += 1
25        # Store the texts by category
26        samples.setdefault(category, []).append((short_text, text))
27    return histogram, samples
28
29 # Classify the expanded tweets using MonkeyLearn, return the histogram
30 tweets_histogram, tweets_categorized = category_histogram(expanded_tweets, tweets_english)
31
32 # Classify the expanded bios of the followed users using MonkeyLearn, return the histogram
33 descriptions_histogram, descriptions_categorized = category_histogram(expanded_descriptions,
34 descriptions_english)

```

Figure 3. Tweet Segregation code