Sentiment Analysis Project using Hadoop

- This File contains the code of My Sentiment Analysis Project that I made at Hadoop map-reduce For the Big Data Subject.
- Code is consisting of Three-classes:

1. Mapper
2. Reducer
3. Driver

- To Run this code, Create a java project at eclipse at cloudera and then create 3 classes and take this code into them then create a .jar file to manage you from running it at Hadoop terminal.
- Then, Put the files at Hadoop fs, files of positive ,negative ,stop and input file and then run jar file and these files and determine name of output.
- The final step prints the output of running.

1. Mapper Class Code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class SentimentMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        private Set<String> positiveWords = new HashSet<String>();
          private Set<String> negativeWords = new HashSet<String>();
          private Set<String> stopWords = new HashSet<String>();

          @Override
          protected void setup(Context context) throws IOException, InterruptedException {
            // Load positive, negative and stop word lists from input files
            Configuration conf = context.getConfiguration();
            FileSystem fs = FileSystem.get(conf);
            Path posWordsFile = new Path(conf.get("positiveWordsFile"));
            Path negWordsFile = new Path(conf.get("negativeWordsFile"));
            Path stopWordsFile = new Path(conf.get("stopWordsFile"));
            positiveWords = loadWordsFromFile(fs, posWordsFile);
            negativeWords = loadWordsFromFile(fs, negWordsFile);
            stopWords = loadWordsFromFile(fs, stopWordsFile);
           }

          @Override
          public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
           String line = value.toString().toLowerCase();
           //creating counters
           int wordCount = 0;
           int goodwords = 0;
           int badwords = 0;
          //int sentimentScore = 0;

           for (String word : line.split("\\s+")) {
                if (!stopWords.contains(word)) {
              if (positiveWords.contains(word)) {
                goodwords++;
                //sentimentScore++;
              }
              else if (negativeWords.contains(word)) {
                badwords++;
              //sentimentScore--;
              }
```

```
                    wordCount++;
                  }

                      }if (wordCount > 0) {

                      context.write(new Text("Positive Words Count  = "), new IntWritable((int)
goodwords));
                                    context.write(new Text("Negative Words Count  = "), new
IntWritable((int) badwords));

                                    double Score = ((double)(goodwords - badwords) / (goodwords +
badwords))* 100;
                            context.write(new Text("The Sentiment score = (" + goodwords + " - " +
badwords + ") / (" + goodwords + " + " + badwords + ")" + "="), new IntWritable((int) Score));

                            //Ratio of Positive
                            double positivityScore = ((double)(goodwords) / (goodwords +
badwords))*100;
                                    context.write(new Text("Positivity score = (" + goodwords + ") / ("
+ goodwords + " + " + badwords + ")" + "="), new IntWritable((int) positivityScore));

                                    //Ratio of Negative
                                    double negativityScore = ((double)(badwords) / (goodwords +
badwords))*100;
                                    context.write(new Text("Negativity score = (" + badwords + ") / ("
+ goodwords + " + " + badwords + ")" + "="), new IntWritable((int) negativityScore));

                                    //if you want to know the count of all words at input file un-hash &
run below line
                                    //context.write(new Text("The Sum of total Words Count  = "), new
IntWritable((int) wordCount));

                                    //NOT Printed draft step
                                    //double sentimentScoreNormalized = ((double) sentimentScore /
wordCount) * 100;
                            //context.write(new Text("Overall Sentiment Score  = "), new
IntWritable((int) sentimentScoreNormalized));
                        }
            }

                  private Set<String> loadWordsFromFile(FileSystem fs, Path filePath) throws
IOException {
                Set<String> words = new HashSet<String>();
                try (BufferedReader br = new BufferedReader(new
InputStreamReader(fs.open(filePath)))) {
                  String line = br.readLine();
                  while (line != null) {
                    words.addAll(Arrays.asList(line.split("\\s+")));
                    line = br.readLine();
                  }
                }
                return words;
              }   }
```

2. Reducer Class Code:

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class SentimentReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

   @Override
   public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
       int totalScore = 0;
       int count = 0;
       for (IntWritable score : values) {
         totalScore += score.get();
         count++;
       }
       if (count > 0) {
          double avgScore = ((double) totalScore / count);
          context.write(key, new IntWritable((int) avgScore));
         }
       }
}
```

3. Driver Class Code:

```
import org.apache.hadoop.conf.Configuration;
//import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class SentimentAnalysis {

   public static void main(String[] args) throws Exception {
      if (args.length != 5) {
         System.err.println("Usage: SentimentAnalysis <pos-words-file> <neg-words-file> <stop-
words-file> <test-file> <output-dir>");
         System.exit(1);
      }

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "Sentiment Analysis");

      // Set JAR class
      job.setJarByClass(SentimentAnalysis.class);

      // Set Mapper class
      job.setMapperClass(SentimentMapper.class);

      // Set Reducer class
      job.setReducerClass(SentimentReducer.class);

    // Set output Key and Value classes
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      // Set input and output paths
      FileInputFormat.addInputPath(job, new Path(args[3]));
      FileOutputFormat.setOutputPath(job, new Path(args[4]));

      // Set the path of the positive, negative, and stop words files
      job.getConfiguration().set("positiveWordsFile", args[0]);
      job.getConfiguration().set("negativeWordsFile", args[1]);
      job.getConfiguration().set("stopWordsFile", args[2]);

      // Run the job and wait for completion
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

The Run Screenshot Result at two files:

- **At simple text file2:**

```
Negative Words Count  =          2
Negativity score = (2) / (3 + 2)=       40
Positive Words Count  =          3
Positivity score = (3) / (3 + 2)=       60
The Sentiment score = (3 - 2) / (3 + 2)=        20
[cloudera@quickstart workspace]$
```

- **At big size text files1:**

```
[cloudera@quickstart workspace]$ hadoop fs -cat output_project_my8/*
Negative Words Count  =          42163
Negativity score = (42163) / (41184 + 42163)=           51
Positive Words Count  =          41184
Positivity score = (41184) / (41184 + 42163)=           49
The Sentiment score = (41184 - 42163) / (41184 + 42163)=     -0.01174
[cloudera@quickstart workspace]$
```