Assignment 3
Sentiment Analysis using Elastic MapReduce

By

Torrencio Vigilante
Cisc 432
The Honourable Patrick Martin

SYSTEM CONFIGURATION

The Table below outlines the configuration of the systems used for the assignment.

Master Node	Number of Core Nodes	Task Instances	Hadoop Version
1 M3.xlarge	3 m1.medium	none	Amazon 2.4.0
1 M3.xlarge	6 m1.medium	none	Amazon 2.4.0
1 M3.xlarge	3 m1.large	none	Amazon 2.4.0

PERFORMANCE METRICES

System	Map Run Time Estimation	Reduce Run Time Estimation	Total Runtime
3 Medium	2 hours 15 minutes	1 hour	2 hour 51 minutes
6 Medium	46 minutes	40 minutes	56 minutes
3 Large	40 Minutes	24 Minutes	42 Minutes

EXPEREINCE AND DISCUSSION OF DATA

When looking at the data, it is clear that as the nodes increase, there is tremendous improvement in computational speeds. What I found amazing is how much concurrency is happening in the medium node. The reduce run time and map run time overlap which is why the total runtime is not the sum of both times. I wish I had tried to use task instances to test whether they had played a huge role in the computational speed. As the nodes increased, there was a clear increase in computational speed. The large nodes seem to increase speed in comparison to both medium setups.

Using the EMR was generally straightforward, though when trying to do more advanced techniques, there was not much documentation. Over all it was a good experience and I had learned a great deal from the exercise.

SENTIMENT ANALYSIS ALGORITHM

Sentiment analysis is very difficult to guarantee over a set of data. After discussing with class mates, the general agreed upon algorithm would be to break each body in the mapper into separate mappings of the format <productID, PositiveOrNegative>. These would be taken and summed in the reducer. This Method works generally, and runs very quickly. My issue with it, is that when someone write a review such as “REALLY

GOOD!" compared to a review of "BAD BAD BAD BAD BAD". In terms of weightings and summing in the reducer phase, the bad review will contribute more to the final weighting.

My original thought to fix this would be to add a secondary key to the output of the mapper such that<productID,MemberID, PositiveOrNegative>, I could not figure out how to add subkeys to the output, so I thought maybe using <productID-MemberID, PositiveOrNegative>. But only being able to reduce once made this a little more challenging.

The method I chose to use was, to do computations in the mapper, would have preferred to have done everything in the reducer, but figured if I'm iterating through the body anyways, I can sum good words and bad words. If there is a good word in the body, I increment some temporary value, if there is a bad word I decrement. If this value is greater than 1 at the end of the body, it is of course a good review. If it is less, it is a bad review (0 is a neutral). One of the three results (positive, negative, neutral) are output with productID in format of <productID,MemberID, PositiveOrNegative>. This means when reduction occurs each distinct review will only count as one towards the total. This may seem like a more costly solution, but I would say the results should be more close to correct. Of course this is not a solution to all reviews, as I think it would struggle with sarcastic reviews and text of more abstract nature.

APPENDIX

Reducer Class:

```
public class reducer extends MapReduceBase implements Reducer<Text, Text, Text,
Text>{

	//In the reduce we collect all nodes mapped with the same key and check
whether the review was
	//negative or positive by summing the values of each

	public void reduce(Text key, Iterator<Text> values, OutputCollector<Text,
Text> output, Reporter reporter) throws IOException {
		String Review= "Neutral";
		int totalScore = 0;

		while (values.hasNext()){
			String val = values.next().toString();
			if (val.equals("positive")){
				totalScore++;
			}else if (val.equals("negative")){
				totalScore--;
			} //neutral do nothing
			//output.collect(key, new Text(val));
		}
		if (totalScore > 0){
			Review = "Positive";
		} else if (totalScore < 0){
			Review = "Negative";
		} //If 0, majority feeling still neutral

		output.collect(key, new Text(Review));

	}
}
```

Mapper Class

```
public class mapper extends MapReduceBase implements Mapper<LongWritable, Text,
Text, Text>{

	private String productID, body;

	//gave up on hash table, though it would be better,
	ArrayList<String> positiveWords = new ArrayList<String>();
	ArrayList<String> negativeWords = new ArrayList<String>();
```

```
	public void map(LongWritable key, Text value, OutputCollector<Text, Text>
output, Reporter report) throws IOException {
		int score = 0;
		String feeling = "nothing";
		String word;

		String review = value.toString();
		String splitreview[] = review.split("\t");
		productID = splitreview[1];
		if(splitreview.length>7&&splitreview[7]!=null){//had some null
bodies, some class mates suggested that there are reviews

				// that are not long enough.
		body = splitreview[7];
		}
		else
			body="";



		StringTokenizer breakingApart = new StringTokenizer(body);

		while(breakingApart.hasMoreTokens()){
			//If word in body exists in hash table of positive and negative
words we alter the score accordingly
			word = breakingApart.nextToken();
			word = word.replaceAll("[^A-Za-z]","");
			if (positiveWords.contains(word)){
				score += 1;
				feeling = "got atleast one word in array list";
			} else if (negativeWords.contains(word)){
				score -= 1;
				feeling = "got atleast one word in array list";
			}
		}

		//output.collect(new Text(productID), new Text(feeling));

		if (score > 0){
			feeling = "positive";
		} else if (score < 0){
			feeling = "negative";
		} else {
			feeling = "neutral";
		}

		//Output key-value pair of (ProductID, feeling)
		output.collect(new Text(productID), new Text(feeling));


	}

	public void configure(JobConf job){
```

```
                try {
                //          FileSystem fs =
FileSystem.get(URI.create("https://s3.amazonaws.com/data432/positive.txt"),  confPos);
                //i couldnt get this to work, so i just uploaded files
                        FileSystem fs = FileSystem.get(new URI("s3://8tjv/"),job);

                        String path1 = job.get("path1");
                        Path p1 = new Path(path1);
                    //Read positive file from uri in path1
                    BufferedReader cacheReader = new BufferedReader(new
InputStreamReader(fs.open(p1)));
                    // now one can use BufferedReader's readLine() to read data
                    String line;
                    while ((line = cacheReader.readLine()) != null){
                        //this was useful from the lectures
                                positiveWords.add(line.trim());

                    }

                    String path2 = job.get("path2");
                        Path p2 = new Path(path2);
                    //Read negative.txt
                    BufferedReader cacheReader2 = new BufferedReader(new
InputStreamReader(fs.open(p2)));
                    // now one can use BufferedReader's readLine() to read data
                    while ((line = cacheReader2.readLine()) != null){

                                negativeWords.add(line.trim());

                    }
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (URISyntaxException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
}
```

Driver Class

```
public class driver extends Configured implements Tool
{       //this class was pretty much from the lecture notes!
        public int run(String[] args) throws Exception{

                //get configuration parameters
                JobConf conf = new JobConf(getConf(), driver.class);
                conf.setJobName("Unreduced Scores For Interest Words For Every
Review");

                //setting key value types for mapper and reducer outputs
```

```
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(Text.class);

                //specifying the custom reducer class
                conf.setReducerClass(reducer.class);
                conf.setMapperClass(mapper.class);

                //Specifying the input directories(@ runtime) and Mappers
independently for inputs from multiple sources
                FileInputFormat.addInputPath(conf, new Path(args[2]));

                //Specifying the output directory @ runtime
                FileOutputFormat.setOutputPath(conf, new Path(args[3]));

                conf.set("path1",args[0]);
                conf.set("path2",args[1]);

                JobClient.runJob(conf);

                FileSystem fs = FileSystem.get(new URI("s3://8tjv/"),conf);
                FileUtil.copyMerge(fs, new Path(args[3]), fs, new Path(args[3]), false,
conf,"");

                return 0;
        }

        public static void main(String[] args) throws Exception{
                int res = ToolRunner.run(new Configuration(),  new driver(), args);
                System.exit(res);
        }

}
```

Program Output (This is the Test file, as the larger is kinda big, I have attached it to moodle though)

0312976275	Positive	B000004BPD	Positive
0380812010	Positive	B000005JAC	Positive
0394756444	Negative	B000005X1J	Positive
0425181685	Positive	B00000JZXJ	Positive
0441003745	Positive	B00000K19E	Positive
0505522659	Positive	B000024KBA	Positive
050552337X	Positive	B00003CWPL	Positive
0553574272	Negative	B00003CX9S	Positive
0553580132	Positive	B00003CXE7	Positive
0691008752	Positive	B00003CXHM	Positive
0783228473	Positive	B00004Z4WX	Positive
0821769820	Positive	B000059H9C	Positive
1575665700	Positive	B00005O1GE	Negative
1885173172	Negative	B000066EX9	Positive
1885478461	Negative	B00006FDQR	Positive
6305078564	Positive	B00006FDR8	Positive
B00000153R	Positive	B000077VQC	Positive
B00000276F	Positive	B00008DDVV	Neutral
B000002P4L	Positive	B00008J4P5	Positive

B000096I8G	Negative
B00009KO14	Positive
B00019RD1Y	Positive
B00028HBKM	Positive
B0002GMSC0	Positive
B00062IVM6	Positive
B00064LJVE	Negative
B0007RT9LC	Positive