

PROJECT 2 REPORT

Team Number:9

Done by:

- Nageswara Rao Nandigam(nrnxb9)
- Revanth Chakilam(rcww4)
- Syed Moin(slrp3)
- Devender Sarda(dspc8)

Project Tasks:

• **Main Requirements:**

- Using the collection of tweets from Project 1 (or collect a new set), implement MapReduce programs to determine the vocabulary uniqueness of your dataset:
 - M/R: Find the list of words that have duplicates in the tweets' text.
 - M/R: Find the list of words that are unique in the tweets' text.
 - Store the lists in two text files: dups.txt and uniqs.txt
 - Print the ratio of the number of unique words to the number of words with duplicates.

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

1) Implement MapReduce programs to determine the vocabulary uniqueness of your dataset

Implemented the MapReduce program in Hadoop by adding the JAR libraries of Hadoop framework and determined uniqueness and duplicate words from text data of each tweet by passing each tweet as JSON object.

```
URI url=new URI("hdfs://localhost:9000"); //-----> (url where hdfs located)
Configuration con = new Configuration();
FileSystem f=FileSystem.get(url,con);
JobConf conf = new JobConf(unique.class);
conf.setJobName("wordcount");

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

conf.setMapperClass(Map.class);
conf.setCombinerClass(Reduce.class);
conf.setReducerClass(Reduce.class);

conf.setInputFormat(TextInputFormat.class);
conf.setOutputFormat(TextOutputFormat.class);
FileInputFormat.setInputPaths(conf,f.makeQualified(new Path("/user/project/input/")));
FileOutputFormat.setOutputPath(conf, f.makeQualified(new Path("/user/project/Output/")));

JobClient.runJob(conf);
```

Output:

```
2017-04-08 16:37:39,793 INFO [main] mapreduce.LocalJobRunner (LocalJobRunner.java:runTasks(1450)) - reduce task executor complete.
2017-04-08 16:38:00,314 INFO [main] mapreduce.Job (Job.java:monitorAndPrintJob(1367)) - map 100% reduce 100%
2017-04-08 16:38:00,316 INFO [main] mapreduce.Job (Job.java:monitorAndPrintJob(1378)) - Job job_local930691498_0001 completed successfully
2017-04-08 16:38:00,346 INFO [main] mapreduce.Job (Job.java:monitorAndPrintJob(1385)) - Counters: 35
File System Counters
  FILE: Number of bytes read=1218773
  FILE: Number of bytes written=3027330
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=533764944
  HDFS: Number of bytes written=232051
  HDFS: Number of read operations=22
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=5
Map-Reduce Framework
  Map input records=46411
  Map output records=464657
  Map output bytes=4493301
  Map output materialized bytes=608638
  Input split bytes=216
  Combine input records=464657
  Combine output records=38278
  Reduce input groups=19139
  Reduce shuffle bytes=608638
  Reduce input records=38278
  Reduce output records=19139
  Spilled Records=76556
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=227
  Total committed heap usage (bytes)=2052587520
```

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

Map function:

From the input file which is in Hadoop, we are reading one each tweet as JSON object and passing to map function and by JSON parser we are getting text data of tweet. In turn that we are passing to StringTokenizer to get each word from that tweet text data and storing temporary collection as word as key and 1 as value.

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        String line = value.toString();
        JSONObject object=new JSONObject(line);
        if(object.has("text")){
            String str=object.getString("text");
            StringTokenizer tokenizer = new StringTokenizer(str);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken().toLowerCase().replaceAll("\\\\W", ""));
                output.collect(word, one);
            }
        }
    }
}
```

Reduce function:

In reduce function, we are sorting and combing each intermediate collection <key, value> pair as one <individual key, sum(values)>, So we will get basically word count for each word.

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
    }
}
```

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

2) M/R: Find the list of words that have duplicates in the tweets' text.

After getting the word count in reduce function, we are filtering the tweets which are sum>1 as duplicate words.

Duplicate.txt file:

```
rep      10
reparacin      5
reparado      4
repbetoourourke      5
republica      5
repealandreplace      6
repeat      5
repeatedly      5
repel      5
repent      6
repercus      5
repito      5
replaced      5
replacement      10
replies      5
reply      47
replying      6
report      30
reporters      5
reporteya      6
reportovzla      5
repostedit      5
representation      5
reproducciones      6
reproduo      5
republican      6
republicans      6
republikaonline      6
republish      5
reputao      5
request      5
requested      10
```

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

3) M/R: Find the list of words that are unique in the tweets' text.

After getting the word count in reduce function, we are filtering the tweets which are sum==1 as duplicate words.

Unique.txt:

```
dalga 1
danahoule 1
dancsoe 1
danielvfreitas 1
dank 1
davidradsick 1
dchirants 1
dcla 1
deiecek 1
dein 1
dellavedova 1
demanded 1
deploying 1
descansa 1
deslocada 1
detailnya 1
developers 1
devlet 1
diamond 1
dik 1
dims 1
diversity 1
doce 1
doida 1
dominickbischo1 1
donnabee511 1
donnadipity 1
douchebag 1
draymond 1
dto 1
dwrightfdtn 1
eats 1
ederim 1
editar 1
edmonearth 1
elbakoush11 1
electrical 1
eleonor 1
```

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

4) Store the lists in two text files: dups.txt and uniqs.txt

Storing it in local file system as duplicate words in duplicate.txt and unique words in unique.txt

```
        if(sum==1)
        {
            output.collect(key, new IntWritable(sum));
            FileWriter fw= new FileWriter("/home/nag/Desktop/Pb/unique.txt",true);
            BufferedWriter bw=new BufferedWriter(fw);
            bw.write(key+" "+sum+"\n");
            bw.flush();
            bw.close();
            u++;
        }
        else
        {
            output.collect(key, new IntWritable(sum));
            FileWriter fw= new FileWriter("/home/nag/Desktop/Pb/duplicate.txt",true);
            BufferedWriter bw=new BufferedWriter(fw);
            bw.write(key+" "+sum+"\n");
            bw.flush();
            bw.close();
            d++;
        }
    }
```

5) Print the ratio of the number of unique words to the number of words with duplicates.

While storing in filesystem I am counting no of words writing in each file and storing in 2 variables (one for uniqueness words count, one for duplicate words count)

After reduce function done, I am using those variable values for finding ration of uniqueness words: duplicate words

```
.....
System.out.println("\n unique words : duplicate words ratio is"+ " "+1 :"+d/u);
}
```

Output:

```
unique words : duplicate words ratio is 1 :76
```

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

• Extra Requirement:

- Implement a MapReduce program to determine the best time to post a tweet.
 - Propose the metric/criterion of your choice based on the tweet JSON format.
 - Run your program and return the top ten best times to post a tweet on twitter.

1) Propose the metric/criterion of your choice based on the tweet JSON format.

I have taken consideration of only retweeted tweets and the time when it was retweeted. I took 2 hours of window for each day as criteria to calculate the no of retweets which are falling on that window. For that I am taking tweets which has retweeted_status and created_date when it was posted. By taking the date and time I am filtering the tweets of which are between 2 hours' window. i.e. 8 am to 10 am and etc. like that I put the 12 conditions and doing map reduce job to count the tweets for each interval. So, will take 2 hours' interval which has maximum no of tweets. That implies if we put post on that window then there is high probability of getting maximum retweets on that window. So, I thought that is best time to post a tweet.

2) Run your program and return the top ten best times to post a tweet on twitter.

- a) Reading the JSON data from tweets file and parsing the each tweet as JSON object and getting created_date object to get the tweet posted date.

```
br=new BufferedReader(new FileReader("C:/Users/VenkatNag/Desktop/twitter_1L.json"));
StringBuilder str=new StringBuilder();
String line=br.readLine();
while(line!=null)
{
    str.append(line);
    line=br.readLine();
}

result=str.toString();
str.delete(0, str.length());

JSONObject object=new JSONObject(result);
if(object.has("retweeted_status"))
{
    JSONObject o=object.getJSONObject("retweeted_status");
    String s=o.getString("created_at");
```

COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

- b) Passing each retweeted tweet date to Map function and adding the interval window condition to divide the tweets and adding to the collections.

Example like below

```
String day=s.substring(0,3);
int n=Integer.parseInt(s.substring(11, 13));
if(n>=00 &n<02)
{
    if(t.containsKey(day+" between 0 & 2"))
    {
        int y=t.get(day+" between 0 & 2");

        t.put(day+" between 0 & 2",y+1);
    }
    else
    {
        t.put(day+" between 0 & 2",1);
    }
}
else if(n>=02 &n<04)
{
    if(t.containsKey(day+" between 2 & 4"))
    {
        int y=t.get(day+" between 2 & 4");

        t.put(day+" between 2 & 4",y+1);
    }
    else
    {
        t.put(day+" between 2 & 4",1);
    }
}
```

- c) In reduce function, combining the result and emitting the total count for tweets which are having particular time 2 hours window and displaying the top ten list which has maximum no of tweets. That implies will get the top ten best time to post the tweets.

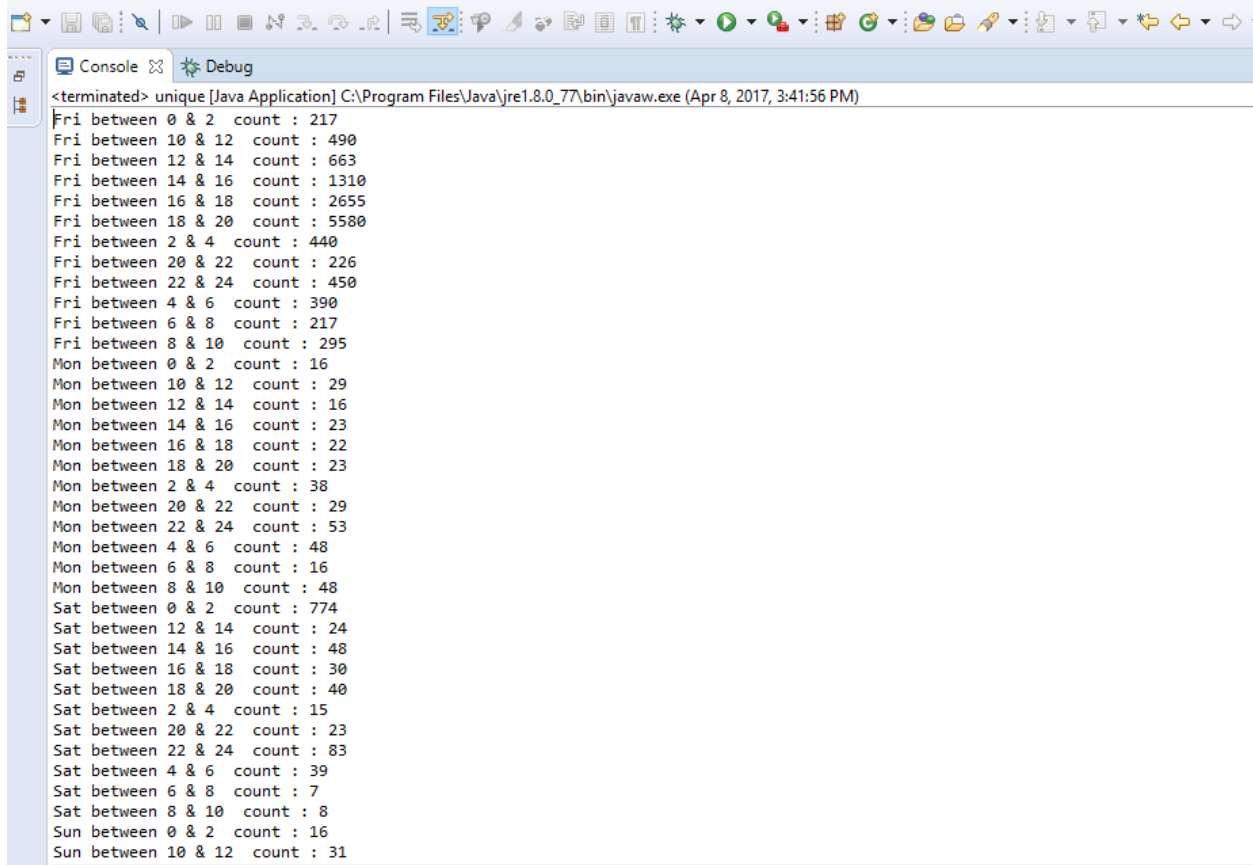
```
System.out.println("Top ten best time to post the tweets \n");
int k=1;
while(k<=10)
{
    Entry<Integer,String> m = (Entry<Integer, String>)tr.lastEntry();

    System.out.println("tweets on "+m.getValue()+" is : "+m.getKey());
    k++;
    tr.remove(m.getKey());
}
```


COMP-SCI 5540 Principles of Big Data Management
University of Missouri-Kansas City
Department of Computer Science and Electrical Engineering

Result:

No of tweets which are tweeted between 2 hours interval on each day



```
<terminated> unique [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 8, 2017, 3:41:56 PM)
Fri between 0 & 2 count : 217
Fri between 10 & 12 count : 490
Fri between 12 & 14 count : 663
Fri between 14 & 16 count : 1310
Fri between 16 & 18 count : 2655
Fri between 18 & 20 count : 5580
Fri between 2 & 4 count : 440
Fri between 20 & 22 count : 226
Fri between 22 & 24 count : 450
Fri between 4 & 6 count : 390
Fri between 6 & 8 count : 217
Fri between 8 & 10 count : 295
Mon between 0 & 2 count : 16
Mon between 10 & 12 count : 29
Mon between 12 & 14 count : 16
Mon between 14 & 16 count : 23
Mon between 16 & 18 count : 22
Mon between 18 & 20 count : 23
Mon between 2 & 4 count : 38
Mon between 20 & 22 count : 29
Mon between 22 & 24 count : 53
Mon between 4 & 6 count : 48
Mon between 6 & 8 count : 16
Mon between 8 & 10 count : 48
Sat between 0 & 2 count : 774
Sat between 12 & 14 count : 24
Sat between 14 & 16 count : 48
Sat between 16 & 18 count : 30
Sat between 18 & 20 count : 40
Sat between 2 & 4 count : 15
Sat between 20 & 22 count : 23
Sat between 22 & 24 count : 83
Sat between 4 & 6 count : 39
Sat between 6 & 8 count : 7
Sat between 8 & 10 count : 8
Sun between 0 & 2 count : 16
Sun between 10 & 12 count : 31
```

From above retrieved top ten list (which are top ten best time to post) based on tweets count



```
Top ten best time to post the tweets

Fri between 18 & 20
Fri between 16 & 18
Fri between 14 & 16
Sat between 0 & 2
Fri between 12 & 14
Fri between 10 & 12
Fri between 22 & 24
Fri between 2 & 4
Fri between 4 & 6
Thu between 20 & 22
```

COMP-SCI 5540 Principles of Big Data Management
University of Missouri-Kansas City
Department of Computer Science and Electrical Engineering

Project Contribution:

