Big Data Analytics (2CS702)

Practical 5	
Rollno: 19BCE055	Name: TEJAS DOBARIYA
Division: A	Batch: E1

AIM : Apply MapReduce algorithms to find phrase frequency from given dataset. • Prepare a report to guide design of mapper and reducer.

CODE :

```
package phrasefreq;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.fs.Path;

public class phrasefreq {
	static int phrase = 0;
	public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>
{

		public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

			String line = value.toString();
```

```
                        String[] tokens = line.split("[ ,.;!?]+");

                        for(int i = 0; i < tokens.length; i+=phrase) {
                                String s = tokens[i].trim();
                                for(int j = i+1; j < Integer.min(i+phrase,tokens.length);
j++) {
                                        s = s+" "+tokens[j].toLowerCase();
                                }

                                context.write(new Text(s), new IntWritable(1));
                        }
                }

        }

        public static class Reduce extends Reducer<Text, IntWritable, Text,
IntWritable> {

                public void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
                        int freq = 0;
                        for(IntWritable num:values) {
                                freq++;
                        }

                        context.write(key, new IntWritable(freq));
                }

                public void cleanup(Context context) throws IOException,
InterruptedException{
                        context.write(new Text("--->Phrase Frequency for Phrase
Length:"), new IntWritable(phrasefreq.phrase));
                }
        }

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

                Configuration conf = new Configuration();

                Job job = Job.getInstance(conf, "LargestNumber");

                job.setJarByClass(phrasefreq.class);
                job.setMapperClass(Map.class);
                job.setReducerClass(Reduce.class);

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

                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);

                Path outputPath = new Path(args[1]);

                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
		try {
		phrase = Integer.parseInt(args[2]);
		} catch(Exception e) {
			phrase = 1;
		}

		outputPath.getFileSystem(conf).delete(outputPath, true);

		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}

}
```

INPUT FILE

file5.txt - Notepad

File Edit Format View Help

```
Mapreduce is useful paradigm to handle big data.
to handle big data, files are split into many parts
mapreduce will handle carefully, each and every file provided to different mapper
mapping is the first, mapreduce will handle carefully the mapper part
by applying business logic. each and every file will be merged now to handle big data.
output will be sorted and then shuffled then provided to reducer part?
provided to reducer part, reducer will handle merging part
```

SCREENSHOTS

```
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Output Format Counters
                Bytes Written=383
2022-10-21 12:15:28,186 INFO mapred.LocalJobRunner: Finishing task: attempt_local557747114_0001_r_000000_0
2022-10-21 12:15:28,188 INFO mapred.LocalJobRunner: reduce task executor complete.
2022-10-21 12:15:28,246 INFO mapreduce.Job:  map 100% reduce 100%
2022-10-21 12:15:29,248 INFO mapreduce.Job: Job job_local557747114_0001 completed successfully
2022-10-21 12:15:29,273 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=11376
                FILE: Number of bytes written=1055448
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=952
                HDFS: Number of bytes written=383
                HDFS: Number of read operations=15
```

Browse Directory

/bda/op5 Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	tejas	supergroup	0 B	Oct 21 12:15	3	128 MB	_SUCCESS
-rw-r--r--	tejas	supergroup	383 B	Oct 21 12:15	3	128 MB	part-r-00000

Showing 1 to 2 of 2 entries Previous 1 Next

Hadoop, 2019.

OUTPUT

part-r-00000 (1) - Notepad

File Edit Format View Help

```
and     3
applying        1
are     1
be      2
big     3
business        1
by      1
carefully       2
data    3
different       1
each    2
every   2
file    2
files   1
first   1
handle  6
into    1
is      2
logic   1
many    1
mapper  2
mapping 1
mapreduce       2
merged  1
merging 1
now     1
output  1
paradigm        1
part    4
parts   1
provided        3
reducer 3
shuffled        1
sorted  1
split   1
the     2
then    2
to      6
useful  1
will    5
--->Phrase Frequency for Phrase Length: 1
```