Assignment

1. Write a code in MapReduce that given a text file counts for each word w the number of occurrences of w in that text file. To this purpose, fill appropriately the map and reduce function included at the end of this handout. Compile your code and create a jar file.

2. A stemmer is a software that extracts from each word its "root". So for example for the words "falling", "faller", "fall" it returns "fall". Write a simple stemmer that removes from each word the strings "er","ing","ed" if they occur at the end of the string. Then write a code in MapReduce that counts the number of occurrences of each word after stemming. For example if in your document there are 4 occurrences of "falling" and 2 of "faller" then your code must produce in output: fall 6. To this end you could use the substring() function in Java.

3. (Extra) Write a MapReduce code to compute the multiplication between a matrix and a vector as well the multiplication between two matrices A and B with the same number of columns and rows.

Once you completed the assignment send it via email to sozio@telecom-paristech.fr with the following subject: INF347-MapReduce.

****

Some info.

hadoop fs -copyFromLocal -to copy from the local machine to the distributed filesystem
-ls -to list the content of a directory in the distributed file system
-text -to visualize the content of a text file

To run the file into hadoop

javac *.java //compile your code
jar cvf mapreduce.jar mapreduce // create a jar file called map reduce.jar from the directory mapreduce
hadoop jar mapreduce.jar mapreduce.MapredDriver books out
//execute the jar file, books is the input file in the distributed filesystem while out is the output file. mapreduce.MapredDriver is the name of the class

****

****file MapredDriver.java****

```
package mapreduce;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class MapredDriver extends Configured implements Tool
{
      public int run(String[] args) throws Exception
      {
            System.out.println("Creating job...");
            Configuration conf = getConf();
            conf.set("mapred.map.tasks","5"); //number of mappers
            conf.set("mapred.reduce.tasks","2"); //number od reducers
            Job job = new Job(conf, "mapred_example");

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

      job.setMapperClass(MapperExample.class);
      job.setReducerClass(ReducerExample.class);

      job.setJarByClass(MapredDriver.class);

      job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(IntWritable.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(LongWritable.class);

      return job.waitForCompletion(false) ? 0 : 1;
      }

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

****file MapperExample.java****

```
package mapreduce;

import java.io.IOException;

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 MapperExample extends Mapper<LongWritable, Text, Text, IntWritable>
{

      public void map(LongWritable key, Text value, Context context)  throws IOException, InterruptedException
      {
            //mapper receives in input a <key,value> pair and produce one or many <key,value> pairs
```

```
            //     in this case key is a string and value is an integer, use context.write() to produce pairs in output
            // example context.write(new Text(w),new IntWritable(1))
            // for separating a document into a list of words you can use string.split(sep)where string is a string object
        }
}
```

****file ReducerExample.java****

```
package mapreduce;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class ReducerExample extends Reducer<Text, IntWritable, Text, LongWritable>
{

    public void reduce(Text key, Iterable<IntWritable> values, Context context)  throws IOException, InterruptedException{

            //reduce receives in input a key and a list of values and produces in output a <key,value> pair        //to produce
in output a key, value pair use context.write(key,new LongWritable(total))        //values is an iterator, in java we can read
elements in the iterator as follows: for(IntWritable vals:values){ vals.get() }
    }
}
```