Mining Massive Datastructures

Assignment : 1

Assigned by
Dr. Subrat K Dash
Associate Professor - Department of CSE

The LNM Institute of Information Technology, Jaipur

March 14, 2021

Shivam Saraswat	18ucc167
Manas Singh	18ucs016
Gaurav Gupta	18ucs214

Contents

0.1 Setup and Installation

0.1.1 Installing Hadoop

Platform: Linux kali 5.10.0-kali3-amd64 #1 SMP Debian 5.10.13-1kali1 (2021-02-08) x86_64 GNU/Linux

We set up a single node cluster.
Java JDK 15 was a dependency already met by our system.
Next, we did not follow the documentation because we intended to use the IntelliJ IDE. The next section talks about setting up IntelliJ for Hadoop projects.

0.1.2 IntelliJ *for* Hadoop

Step 1 : Create a new project and add the following code into a new class **WordCount.java**.

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

        public static class TokenizerMapper
                extends Mapper<Object, Text, Text, IntWritable>{

                private final static IntWritable one = new IntWritable(1);
                private Text word = new Text();

                public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
                        StringTokenizer itr = new StringTokenizer(value.toString());
                        while (itr.hasMoreTokens()) {
                                word.set(itr.nextToken());
                                context.write(word, one);
                        }
                }
        }

        public static class IntSumReducer
                extends Reducer<Text,IntWritable,Text,IntWritable> {
                private IntWritable result = new IntWritable();

                public void reduce(Text key, Iterable<IntWritable> values,
                                   Context context
                ) throws IOException, InterruptedException {
                        int sum = 0;
                        for (IntWritable val : values) {
```

```
                        sum += val.get();
                    }
                    result.set(sum);
                    context.write(key, result);
                }
        }

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                Job job = Job.getInstance(conf, "word count");
                job.setJarByClass(WordCount.class);
                job.setMapperClass(TokenizerMapper.class);
                job.setCombinerClass(IntSumReducer.class);
                job.setReducerClass(IntSumReducer.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

Step 2 : Go to File > Project Structure > Dependencies > +, and click on **JARs or Directories** option.

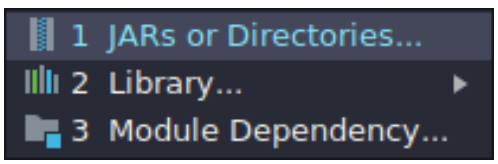

Figure 1: Select JARs and Directories

Step 3 : Next Go to Run > Edit Configurations... > Application > +, and add arguments in the following format.

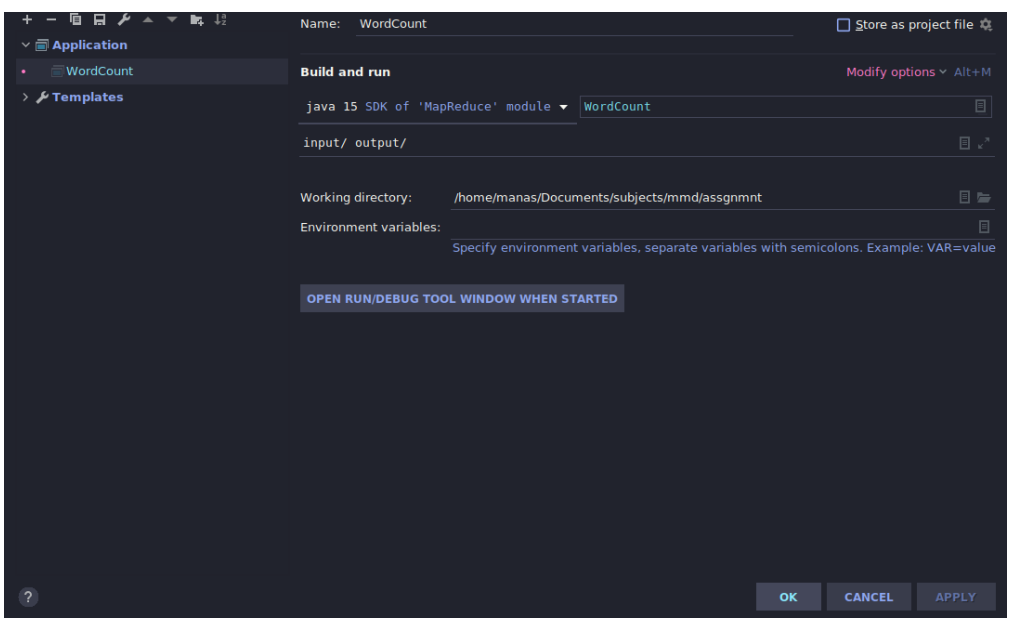

Figure 2: Program Arguments

Step 4 : Go to *share/hadoop* and select all the files by holding down ⇧.

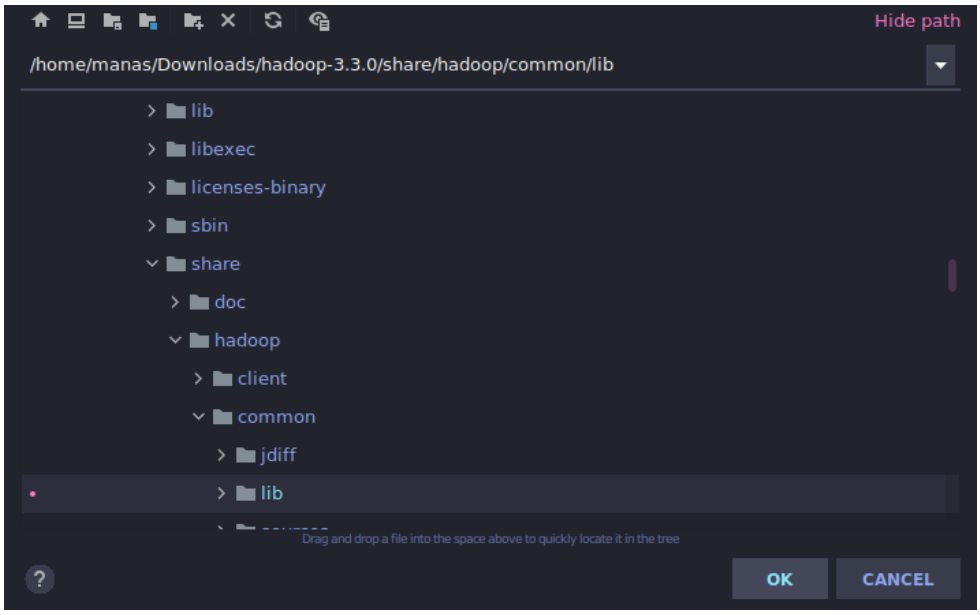

Figure 3: share/hadoop

Step 5 : Go to *share/hadoop/common* and select the **lib** directory.

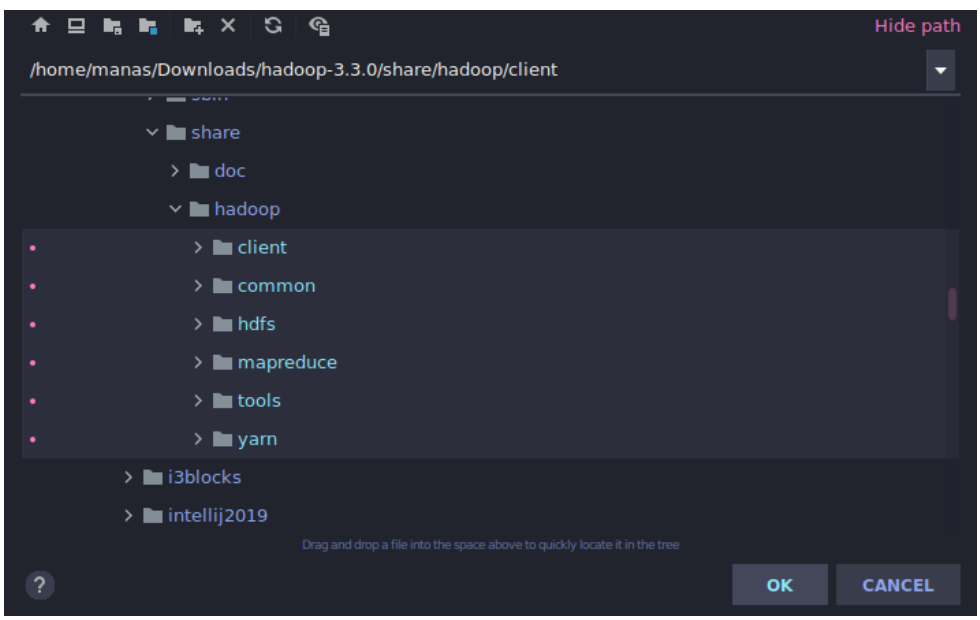

Figure 4: share/hadoop/common

Step 6 : If you did everything right, it will look something like this:

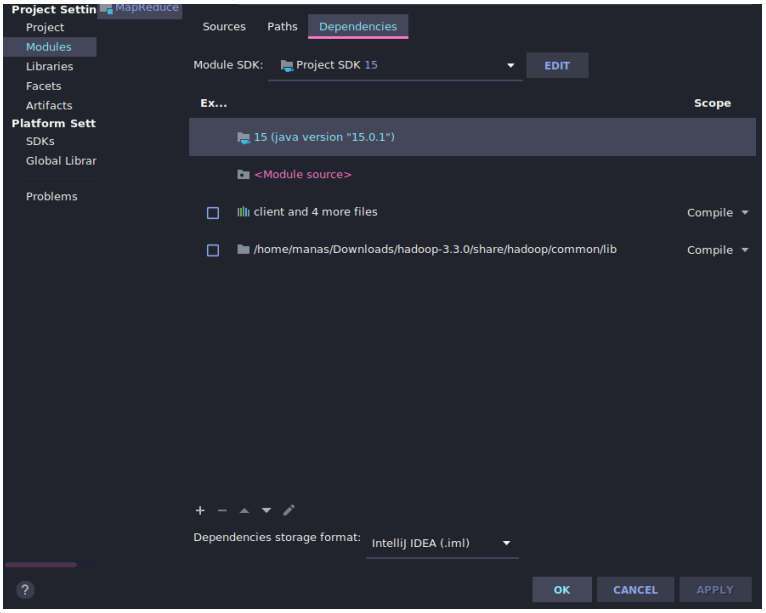

Figure 5: Added dependencies

Step 7 : Copy your text file to **input** directory. In our case **pride_and_prejudice.txt**.

Step 8 : Add the roll numbers of the team members.

```
cat >> input/pride_and_prejudice.txt
CSE3152 18ucc167
CSE3152 18ucs016
CSE3152 18ucs214
^C
```

0.2 Building and Running

0.2.1 First Build

Go ahead and hit **Build and Run** from the menu bar.
If everything works fine, you will get an output like this:

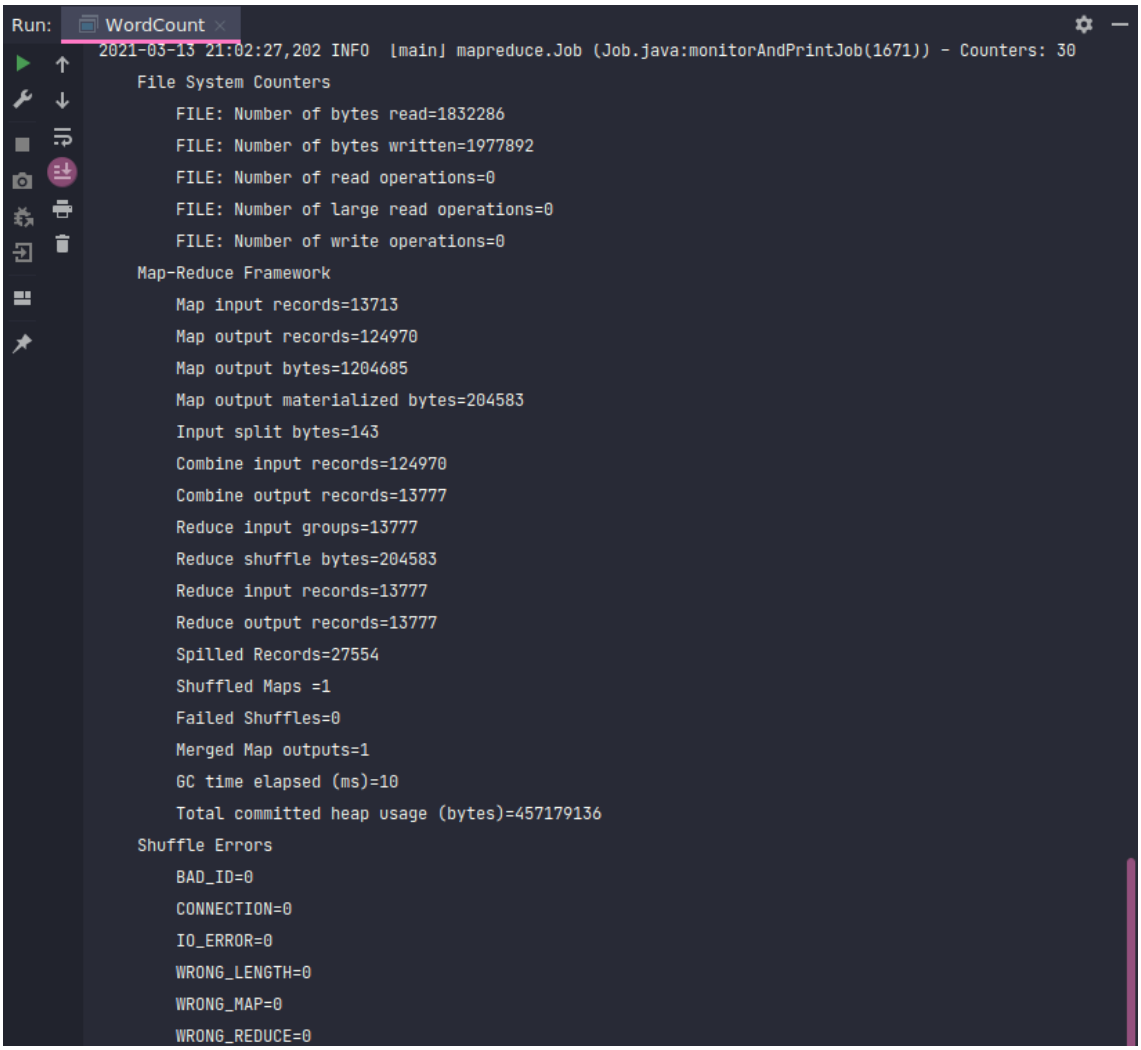

Figure 6: MapReduce statistics

0.2.2 Output

A directory by the name of **output** would be created automatically, each time you run the program. If you want to re-run it, you need to delete the directory, or specify another directory in the arguments by following Step 3. The final file structure would look something like this:

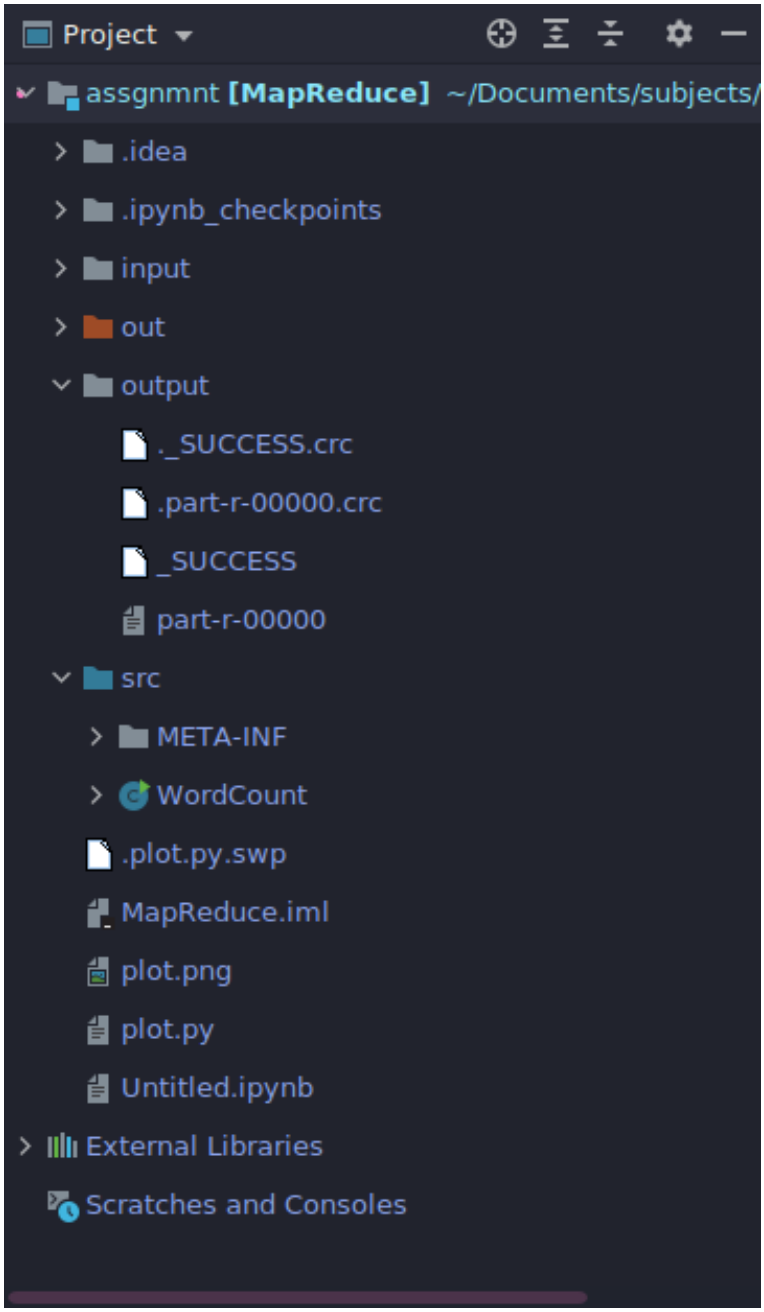

Figure 7: Project File Structure

0.2.3 Artifacts

Now to make this software portable you might want to create an artifact. For that go to File > Project Structure > Artifacts and add a new artifact corresponding to **WordCount.java**. Go to **Build** and click on **Build Artifacts**. From now on you could directly invoke the JAR in the following manner.

```
bin/hadoop jar WordCount.jar input/ output/
```

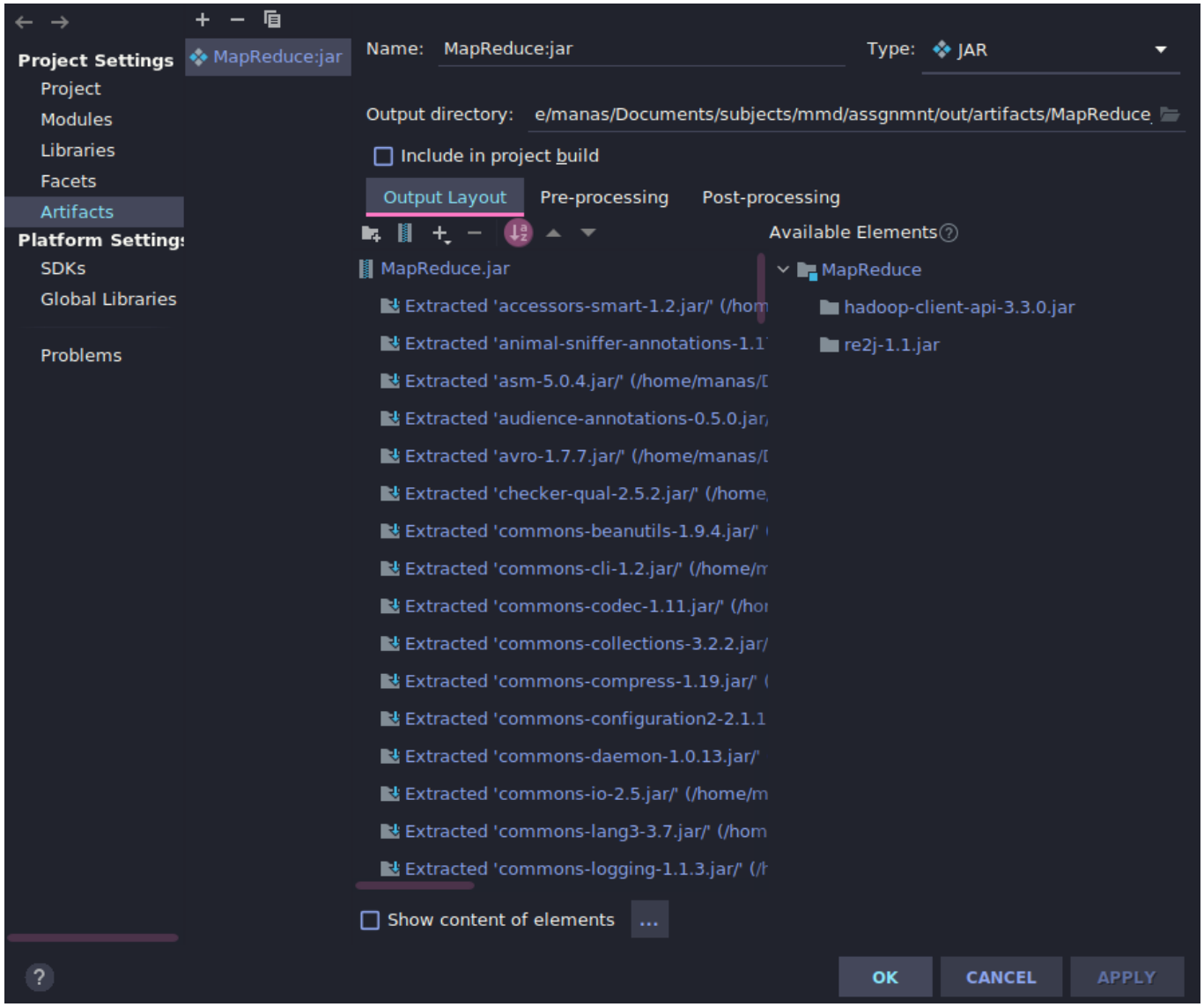

Figure 8: Artifacts Window

0.3 Analyzing and Plots

We analyzed the output file **output/part-r-00000** and made a plot of the last 50 highest frequency words, and added our own roll numbers for authententicity.

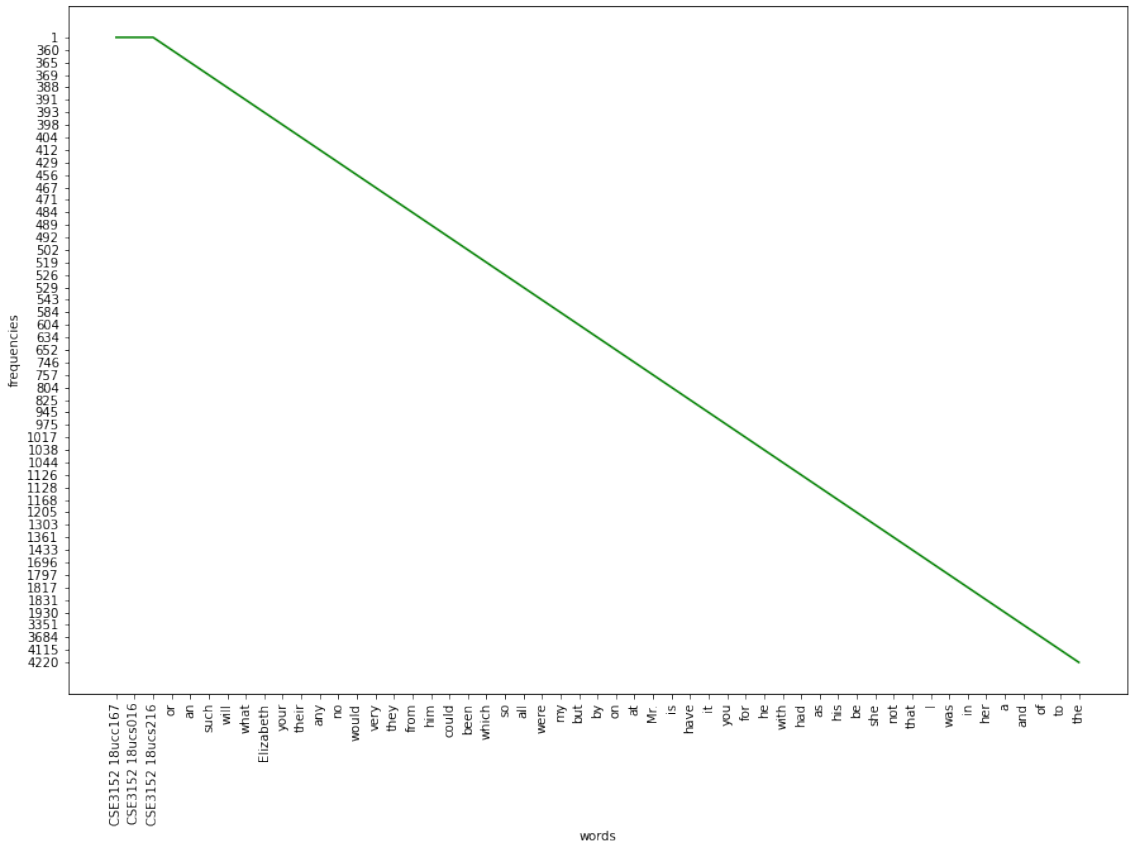

Figure 9: Words vs Frequencies

```
import sys
import matplotlib.pyplot as plt
import seaborn as sns

def plot_graph():
    n = len(sys.argv)
    if n < 2:
         print("File name not specified\n")
         return

    file = open(sys.argv[1], 'r')
    #file = open('./output/part-r-00000', 'r')

    doc = file.read().split('\n')
    data = []
    for x in doc:
        word = x.split('\t')
        if(len(word) < 2): continue
        data.append(word)

    data.sort(key=lambda x : int(x[1]))
```

```
    words = []
    freq = []
    for i in data:
        words.append(i[0])
        freq.append(i[1])

    # The roll nos. were already added in the input data, but still adding them here
    # so that they show up in the graph too!
    roll_list = [
        "CSE3152 18ucc167",
        "CSE3152 18ucs016",
        "CSE3152 18ucs216",
    ]
    roll_freq = ["1", "1", "1"]

    plt.figure(figsize=(15,15))
    plt.xticks(rotation=90)
    plt.xlabel("words")
    plt.ylabel("frequencies")
    g = sns.lineplot(roll_list + words[-50:], roll_freq + freq[-50:], color="green")

    #plt.show()
    g.plot()
    file.close()

if __name__ == '__main__':
    plot_graph()
```

0.4 Files and Source

https://drive.google.com/drive/folders/1bmPB2YIjPwCezrD71AA7GhtOE801Q6Gj?usp=sharing

This report was submitted with all the files itself in ZIP format.

The Github repository will go public post the midnight submission deadline at github.com/kingmanas/MapReduce

Link to pride_and_prejudice.txt : http://gutenberg.org/ebooks/42671

Apache Tutorials:

Setting up Hadoop : https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html
Map Reduce in Hadoop : https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client MapReduceTutorial.html#Example:_WordCount_v1.0]