```java
package com.naufal.selecting_key_twitter;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
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;
import org.apache.hadoop.mapreduce.lib.reduce.IntSumReducer;
/**
 *
 * @author ahmadluky
 * Chain Job Mapreduce
 */
public class Accessibility {
    // reply |akun, is_reply|
    public static class TaskMapper_reply extends Mapper<Object, Text, Text, IntWritable>{
        private Text word = new Text();
        private IntWritable reply = new IntWritable(1);
        private IntWritable reply_zero = new IntWritable(0);
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
            String[] split = value.toString().split(",");
            if (split[1].equalsIgnoreCase("TRUE")) {
                word.set(split[0]);
                context.write(word, reply);
            }else{
                word.set(split[0]);
                context.write(word, reply_zero);
            }
        }
    }
    public static class TaskReducer_reply 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);
        }
    }
    //mention
    public static class TaskMapper_mention extends Mapper<Object, Text, Text, IntWritable>{
        private Text word = new Text();
        private IntWritable mention = new IntWritable(1);
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
            if (!value.equals(null)) {
                String tString = value.toString().replace('"', ' ');
                String tString2 = tString.replaceAll("\\s+","");
                String[] split = tString2.split(",");
                if ( split.length>0) {
                    for (int i = 0; i < split.length; i++) {
                        if (!split[i].equalsIgnoreCase(" ")) {
                            word.set(split[i]);
                            System.out.println(split[i]);
                            context.write(word, mention);
                        }
                    }
                }
            }
        }
    }
    public static class TaskReducer_mention 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);
        }
    }
    // quote
    public static class TaskMapper_quote extends Mapper<Object, Text, Text, IntWritable>{
        private Text word = new Text();
        private IntWritable one = new IntWritable(1);
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
            word.set(value);
            context.write(word, one);
         }
     }
     public static class TaskReducer_quote 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);
         }
     }
    // jumlsh kelas positif
    // jumlah kelas negatif
    public static class TaskMapper_sentimen extends Mapper<Object, Text, Text, IntWritable>{
        private Text word = new Text();
        private IntWritable one = new IntWritable(1);
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
            word.set(value);
            context.write(word, one);
          }
     }
     public static class TaskReducer_sentimen 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);
          }
     }
    // accessibility
    public static class TaskMapper extends Mapper<Object, Text, Text, IntWritable>{
        private Text word = new Text();
        private IntWritable rst_temp = new IntWritable();
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
            String[] split = value.toString().split(";");
            word.set(split[0]);
            int sum = 0;
            if (split.length == 4) {
                sum += Integer.parseInt(split[1]) + Integer.parseInt(split[2]) +
Integer.parseInt(split[3]);
                rst_temp.set(sum);
                context.write(word, rst_temp);
            }
        }
    }
    public static class TaskReducer 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);
        }
    }
```

```
148     // nilai_aksesibilitas = folowing(i)+jml_reply(i)+( (quote(i)+mention(i)) * (jumlah akun pos/jumlah
    akun neg) )
149     // 1. get jml_reply
150     // 2. get quote
151     // 3. get mention
152     // 4. get jumlah akun pos -
153     // 5. get jumlah akun neg -
154     public static void main(String[] args) throws Exception {
155
156         /**
157          * get quote
158          */
159         String quote_in = "data-in/accessibility/quote";
160         String quote_out = "data-out/accessibility/quote";
161         Configuration conf_quote = new Configuration();
162         FileSystem fs_quote = FileSystem.get(conf_quote);
163         if (fs_quote.exists(new Path(quote_in))) {
164             fs_quote.delete(new Path(quote_out), true);
165         }
166         Job job_quote = Job.getInstance(conf_quote, "quote_count");
167         job_quote.setJarByClass(Accessibility.class);
168         job_quote.setMapperClass(TaskMapper_quote.class);
169         job_quote.setCombinerClass(IntSumReducer.class);
170         job_quote.setReducerClass(TaskReducer_quote.class);
171         job_quote.setOutputKeyClass(Text.class);
172         job_quote.setOutputValueClass(IntWritable.class);
173         FileInputFormat.addInputPath(job_quote, new Path(quote_in));
174         FileOutputFormat.setOutputPath(job_quote, new Path(quote_out));
175         job_quote.waitForCompletion(true);
176         if (!job_quote.isSuccessful()) {
177             System.out.println("Job Finding Local " + job_quote.getJobID() + " failed!");
178             System.exit(1);
179         }
180         System.exit(-1);
181         /**
182          * get mention
183          */
184         String mention_in = "data-in/accessibility/mention";
185         String mention_out = "data-out/accessibility/mention";
186         Configuration conf_mention = new Configuration();
187         FileSystem fs_mention = FileSystem.get(conf_mention);
188         if (fs_mention.exists(new Path(mention_in))) {
189             fs_mention.delete(new Path(mention_out), true);
190         }
191         Job job_mention = Job.getInstance(conf_mention, "Reply_count");
192         job_mention.setJarByClass(Accessibility.class);
193         job_mention.setMapperClass(TaskMapper_mention.class);
194         job_mention.setCombinerClass(IntSumReducer.class);
195         job_mention.setReducerClass(TaskReducer_mention.class);
196         job_mention.setOutputKeyClass(Text.class);
197         job_mention.setOutputValueClass(IntWritable.class);
198         FileInputFormat.addInputPath(job_mention, new Path(mention_in));
199         FileOutputFormat.setOutputPath(job_mention, new Path(mention_out));
200         job_mention.waitForCompletion(true);
201         if (!job_mention.isSuccessful()) {
202             System.out.println("Job Finding Local " + job_mention.getJobID() + " failed!");
203             System.exit(1);
204         }
205         System.exit(-1);
206         /**
207          * get jumlah_reply
208          */
209         String jumlah_reply_in = "data-in/accessibility/reply";
210         String jumlah_reply_out = "data-out/accessibility/reply";
211         Configuration conf_reply = new Configuration();
212         FileSystem fs_reply = FileSystem.get(conf_reply);
213         if (fs_reply.exists(new Path(jumlah_reply_in))) {
214             fs_reply.delete(new Path(jumlah_reply_out), true);
215         }
216         Job job_reply = Job.getInstance(conf_reply, "Reply_count");
217         job_reply.setJarByClass(Accessibility.class);
218         job_reply.setMapperClass(TaskMapper_reply.class);
219         job_reply.setCombinerClass(IntSumReducer.class);
220         job_reply.setReducerClass(TaskReducer_reply.class);
221         job_reply.setOutputKeyClass(Text.class);
222         job_reply.setOutputValueClass(IntWritable.class);
223         FileInputFormat.addInputPath(job_reply, new Path(jumlah_reply_in));
224         FileOutputFormat.setOutputPath(job_reply, new Path(jumlah_reply_out));
225         job_reply.waitForCompletion(true);
```

```
226         if (!job_reply.isSuccessful()) {
227           System.out.println("Job Finding Local " + job_reply.getJobID() + " failed!");
228           System.exit(1);
229         }
230         /**
231          * accessibility
232          */
233         String accessibility_in = "data-in/accessibility";
234         String accessibility_out = "data-out/accessibility";
235         Configuration conf = new Configuration();
236         FileSystem fs = FileSystem.get(conf);
237         if (fs.exists(new Path(accessibility_in))) {
238             fs.delete(new Path(accessibility_out), true);
239         }
240         Job job = Job.getInstance(conf, "Accessibility");
241         job.setJarByClass(Accessibility.class);
242         job.setMapperClass(TaskMapper.class);
243         job.setCombinerClass(IntSumReducer.class);
244         job.setReducerClass(TaskReducer.class);
245         job.setOutputKeyClass(Text.class);
246         job.setOutputValueClass(IntWritable.class);
247         FileInputFormat.addInputPath(job, new Path(accessibility_in));
248         FileOutputFormat.setOutputPath(job, new Path(accessibility_out));
249         job.waitForCompletion(true);
250         if (!job.isSuccessful()) {
251           System.out.println("Job Finding Local " + job.getJobID() + " failed!");
252           System.exit(1);
253         }
254     }
255 }
256
```