. **Unique Users in the data set**

package unique\_users;

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 unique\_user {

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());

String validation\_string = word.toString();

StringTokenizer validation = new StringTokenizer(validation\_string, ".");

int count = 0;

while(validation.hasMoreTokens())

{

count += 1;

validation.nextToken();

}

if (count == 4)

{

context.write(word, one);

}

itr.nextToken();

itr.nextToken();

itr.nextToken();

}

}

}

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(unique\_user.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);

}

1. **User Count program**

package user\_frequency\_count;

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 user\_frequency\_count {

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());

String validation\_string = word.toString();

StringTokenizer validation = new StringTokenizer(validation\_string, ".");

int count = 0;

while(validation.hasMoreTokens())

{

count += 1;

validation.nextToken();

}

if (count == 4)

{

context.write(word, one);

}

itr.nextToken();

itr.nextToken();

itr.nextToken();

}

}

}

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(user\_frequency\_count.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); }

}

2. **Extract maximum frequency user** program package maximum\_user\_frequency;

import java.io.DataInput;

import java.io.DataOutput;

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.io.Writable;

import org.apache.hadoop.io.WritableComparable;

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 MaximumUserFrequency{

static class User implements Writable, WritableComparable<User> { private String name = "";

private Integer freq = new Integer(0);

public void setName(String n)

{

name = n;

}

public void setFreq(Integer f)

{

freq = f;

}

public String getName()

{

return name;

}

public Integer getFreq()

{

return freq;

}

@Override

public void readFields(DataInput in) throws IOException { freq = in.readInt();

name = in.readLine();

}

@Override

public void write(DataOutput out) throws IOException { out.writeInt(freq);

out.writeBytes(name);

}

@Override

public int compareTo(User o) {

// TODO Auto-generated method stub

int result = this.freq.compareTo(o.freq);

return result;

}

}

private static User u = new User();

static class UserMaxCountMapper extends Mapper<Object, Text, Text, User> { private Text user = new Text();

@Override

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

user.set("user");

u.setName(value.toString().split(",")[0]);

u.setFreq(Integer.parseInt(value.toString().split(",")[1])); context.write(user, u);

}

}

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

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

result.setFreq(null);

result.setName("");

for (User value : values)

{

if (result.getFreq() == null || (value.getFreq() > result.getFreq())) { result.setFreq(value.getFreq());

result.setName(value.getName()); }

}

key.set(result.getName());

context.write(key, new IntWritable(result.getFreq()));

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

job.setJarByClass(MaximumUserFrequency.class);

job.setJobName("find\_max\_user\_count");

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

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

job.setMapperClass(UserMaxCountMapper.class);

job.setReducerClass(UserMaxCountReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(User.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

}

}

5. **Word Count**

**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);**

**}**

**}**