import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Map.Entry;

import java.util.Set;

import java.util.StringTokenizer;

import java.util.\*;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

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.Mapper.Context;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Assigment\_1 {

public static class MyMapper 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 {

String str=value.toString().toLowerCase();;

str=str.replaceAll("[^a-zA-Z]", " ");

String[] words = str.split(" ");

ArrayList<String> wordsList = new ArrayList<String>();

Set<String> stopWordsSet = new HashSet<String>();

stopWordsSet.add("a");

stopWordsSet.add("an");

stopWordsSet.add("the");

stopWordsSet.add("this");

stopWordsSet.add("and");

stopWordsSet.add("or");

stopWordsSet.add("he");

stopWordsSet.add("she");

stopWordsSet.add("it");

stopWordsSet.add("they");

stopWordsSet.add("that");

stopWordsSet.add("them");

stopWordsSet.add("a");

stopWordsSet.add("b");

stopWordsSet.add("c");

stopWordsSet.add("d");

stopWordsSet.add("e");

stopWordsSet.add("f");

stopWordsSet.add("g");

stopWordsSet.add("h");

stopWordsSet.add("i");

stopWordsSet.add("j");

stopWordsSet.add("k");

stopWordsSet.add("l");

stopWordsSet.add("m");

stopWordsSet.add("n");

stopWordsSet.add("o");

stopWordsSet.add("p");

stopWordsSet.add("q");

stopWordsSet.add("r");

stopWordsSet.add("s");

stopWordsSet.add("t");

stopWordsSet.add("u");

stopWordsSet.add("v");

stopWordsSet.add("w");

stopWordsSet.add("x");

stopWordsSet.add("y");

stopWordsSet.add("z");

for(String word : words)

{

String wordCompare=word;

if(!stopWordsSet.contains(wordCompare))

{

wordsList.add(word);

}

}

String s=new String("");

for (String ns : wordsList){

s=s+" "+ns;

}

StringTokenizer itr = new StringTokenizer(s);

while (itr.hasMoreTokens()) {

String t=itr.nextToken();

word.set(String.valueOf(t.charAt(0)));

context.write(word,new IntWritable(t.length()));

}

}

}

public static class MyReducer

extends Reducer<Text,IntWritable,Text,DoubleWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

double sum = 0;

int count=0;

for (IntWritable val : values) {

sum += val.get();

count++;

}

sum=sum/count;

context.write(new Text(key),new DoubleWritable(sum));

}

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "Assigment");

job.setJarByClass(Assigment\_1.class);

job.setMapperClass(MyMapper.class);

job.setReducerClass(MyReducer.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);

}

}