1. Implement a simple map-reduce job that builds an inverted index on the set of input documents(Hadoop)

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

import java.io.\*;

import java.util.\*;

public class InvertedIndex {

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

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

String docId = ((FileSplit) context.getInputSplit()).getPath().getName();

for (String w : value.toString().toLowerCase().replaceAll("[^a-z0-9 ]", "").split(" "))

if (!w.isEmpty()) context.write(new Text(w), new Text(docId));

}

}

public static class IndexReducer extends Reducer<Text, Text, Text, Text> {

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

TreeSet<String> uniqueDocs = new TreeSet<>();

values.forEach(v -> uniqueDocs.add(v.toString()));

context.write(key, new Text(String.join(", ", uniqueDocs)));

}

}

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

Job job = Job.getInstance(new Configuration(), "Inverted Index");

job.setJarByClass(InvertedIndex.class);

job.setMapperClass(TokenizerMapper.class);

job.setReducerClass(IndexReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}Execution

1. Compile:

hadoop com.sun.tools.javac.Main InvertedIndex.java

jar cf invertedindex.jar InvertedIndex\*.class

1. Run on Hadoop:

hadoop jar invertedindex.jar InvertedIndex /input /output

1. Check the output:

hdfs dfs -cat /output/part-r-00000