**============================================**

**Twitter Data Analysis & User Clustering**

**============================================**

**Project Statement:**

Twitter has vast amount of spontaneous data created by millions of users around the globe. Its popularity around the world in raising voice, trending ideas, communicating with peers makes it a powerful tool. Analysis on that data is a major challenge but it can get you an idea about the current situation in any country.

Data is pre-collected in text files through the Twitter Streaming API (Twitter4j) and the following analytical tasks are run on it in MapReduce:

The data set contains

Text of the tweet.

Screen name of the user who tweeted

Follower count of the user.

Link to dataset: https://drive.google.com/open?id=0B5preb5rCiJPY2ZuSDdGU2NrTms

1. **Word Count:**

**Specification:** To get most trending words, hash tags and mentions.

**Implementation:**

It comprises of two parts:-

Simple logic for Word Count.

Mapper: TokenizerMapper

Reducer: IntSumReducer

**PseudoCode:**

class Mapper  
method Initialize  
H<­ new AssociativeArray  
method Map(docid a, doc d)  
for all term t ∈ doc d do  
H{t} ← H{t} + 1  
method Close  
for all term t ∈ H  
Emit(term t, count H{t})  
class Reducer  
method Reduce(term t, counts [c 1 , c 2 , . . .])  
sum <­ 0  
for all count c ∈ counts [c 1 , c 2 , . . .] do  
sum <­sum + c  
Emit(term t, count sum)

In the second part I do secondary sorting using a combined key (word +  
count).  
Mapper : SortingMapper  
  
Reducer : SortingReducer

1. **Word Co-occurrence:**

**Specification:**

All the possible pairs of word in a tweet are considered in Pairs approach.

A track of occurrences is kept in associative array and this approach is named as Stripes approach.

**Implementation:**

In Pairs approach each word is scanned and emits a ([word, neighbour], count) key- value pair in a single tweet. This only  
give as the regular count so we are also emitting a special key pair as ([Word,\*], count). The special pair would give us the possible  
counts of occurrences of ‘Word’. To ensure that the special pair reaches before regular pair, we make use of  
compareTo() function in the WritableComparable class.

Mapper: PairMapper  
Partitioner: PairPartitioner  
Reducer: PairReducer  
Comparator: PairComparator

While in Stripes approach it becomes easier. It is because  
 we are obtaining the result in an associative array or specifically a  
MapWritable, we can easily calculate the total as well as relative occurrence at the reducer.

Mapper:StripeMapper  
Reducer:StripeReducer  
  
**Use:** To find out the frequency of two concurrent hashtags. You get to know about the trends through hashtags and it’s quite useful to analyze it.

1. **KMeans Clustering:**

**Specification**: Users are clustered based on the number of followers they have, into three sets.

**Use:** This information may be used by advertising agencies.

**Implementation**:

Kmeans clustering algorithm was implemented. Multiple iterations of MapReduce to arrive at a converged value for chosen centroids.

Use of counters to save the state in between iterations.

1. **Shortest Path:**

**Specification:** Modelling users as nodes, each consisting of node id, node distance and the adjacency list with the information of neighbor nodes, the shortest path between 2 nodes.

**Implementation:**

The MapReduce version of the Dijkstra’s Algorithm for shortest paths to all nodes of the given graph from a single source node is used.

I have connected directed graph as input with adjacency list  
indicating neighbors and distance stored alongside. Mapping was done over all nodes.  
A key­-value pair for each neighbor on the node’s adjacency list was emitted by mapper.  
The key value is the node id of the neighbor and value is current distance to node +1. If node n can be reached with distance d then all connected nodes of n can be reached with d+1 distance. After shuffle, reducers have keys corresponding to nodes and distances about all path leading to the node. The reducer then selects shortest and updates the node.

**Code:**

**WordCount**

WordCount.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

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.lib.input.FileInputFormat;

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

import java.io.IOException;

public class WordCount {

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

Configuration conf = new Configuration();

String inputPathWordCount = "C:/Users/param//Desktop/hadoop/FinalProject/data/tweets.txt";

String outputPathWordCount = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/original";

String inputPathWordCountSort = "C:/Users/param//Desktop/hadoop/FinalProject/data/final\_sort";

String outputPathWordCountSort = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/final\_sort";

String inputPathUsersTweet = "C:/Users/param//Desktop/hadoop/FinalProject/data/final\_tweet";

String outputPathUsersTweet = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/final\_tweet";

String outputPathUsersTweetSort = "/output/final\_tweet/sort";

String inputPathHashTags = "C:/Users/param//Desktop/hadoop/FinalProject/data/final\_hash";

String outputPathHashTags = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/final\_hash";

String outputPathHashTagSort = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/final\_hash/sort";

String output = "C:/Users/param//Desktop/hadoop/FinalProject/data/output";

deleteFolder(conf, outputPathWordCount);

deleteFolder(conf, outputPathWordCountSort);

deleteFolder(conf, outputPathUsersTweet);

deleteFolder(conf, outputPathHashTags);

deleteFolder(conf, outputPathUsersTweetSort);

deleteFolder(conf, outputPathHashTagSort);

deleteFolder(conf, inputPathWordCountSort);

deleteFolder(conf, inputPathUsersTweet);

deleteFolder(conf, inputPathHashTags);

deleteFolder(conf, output);

Job job = Job.getInstance(conf);

Job job2 = null, job3 = null, job4 = null;

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(inputPathWordCount));

FileOutputFormat.setOutputPath(job, new Path(outputPathWordCount));

if (job.waitForCompletion(true)) {

job2 = Job.getInstance(conf);

job2.setJarByClass(WordCount.class);

job2.setMapperClass(SortingMapper.class);

job2.setCombinerClass(SortReducer.class);

job2.setReducerClass(SortReducer.class);

job2.setOutputKeyClass(IntWritable.class);

job2.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job2, new Path(outputPathWordCount));

FileOutputFormat.setOutputPath(job2, new Path(outputPathWordCountSort));

}

if (job2.waitForCompletion(true)) {

job3 = Job.getInstance(conf);

job3.setJarByClass(WordCount.class);

job3.setMapperClass(HashTagMapper.class);

job3.setCombinerClass(HashTagReducer.class);

job3.setReducerClass(HashTagReducer.class);

job3.setOutputKeyClass(Text.class);

job3.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job3, new Path(inputPathWordCount));

FileOutputFormat.setOutputPath(job3, new Path(outputPathHashTags));

if(job3.waitForCompletion(true)){

job3 = Job.getInstance(conf);

job3.setJarByClass(WordCount.class);

job3.setMapperClass(SortingMapper.class);

job3.setCombinerClass(SortReducer.class);

job3.setReducerClass(SortReducer.class);

job3.setOutputKeyClass(IntWritable.class);

job3.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job3, new Path(outputPathHashTags));

FileOutputFormat.setOutputPath(job3, new Path(outputPathHashTagSort));

}

}

if (job3.waitForCompletion(true)) {

job4 = Job.getInstance(conf);

job4.setJarByClass(WordCount.class);

job4.setMapperClass(TweetMapper.class);

job4.setCombinerClass(TweetReducer.class);

job4.setReducerClass(TweetReducer.class);

job4.setOutputKeyClass(Text.class);

job4.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job4, new Path(inputPathWordCount));

FileOutputFormat.setOutputPath(job4, new Path(outputPathUsersTweet));

if(job4.waitForCompletion(true)){

job4 = Job.getInstance(conf);

job4.setJarByClass(WordCount.class);

job4.setMapperClass(SortingMapper.class);

job4.setCombinerClass(SortReducer.class);

job4.setReducerClass(SortReducer.class);

job4.setOutputKeyClass(IntWritable.class);

job4.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job4, new Path(outputPathUsersTweet));

FileOutputFormat.setOutputPath(job4, new Path(outputPathUsersTweetSort));

}

System.exit(job4.waitForCompletion(true) ? 0 : 1);

}

}

private static void deleteFolder(Configuration conf, String folderPath)

throws IOException {

FileSystem fs = FileSystem.get(conf);

Path path = new Path(folderPath);

if (fs.exists(path)) {

fs.delete(path, true);

}

}

}

2. TokenizerMapper.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

import java.util.StringTokenizer;

public 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 {

String temp\_url = value.toString();

String temp = temp\_url.replaceAll("http[s]?://t.co/[A-Za-z0-9]+", " ");

temp = temp.replaceAll("[^A-Za-z0-9@#]", " ");

temp = temp.replaceAll("#+", "#");

StringTokenizer itr = new StringTokenizer(temp);

while (itr.hasMoreTokens()) {

String tempStr = itr.nextToken();

if (!StopWords.stopWords.contains(tempStr.toLowerCase())) {

word.set(tempStr);

context.write(word, one);

}

}

}

}

3. StopWords.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import java.util.HashSet;

public class StopWords {

public static HashSet<String> stopWords = new HashSet<String>();

static {

stopWords.add("rt");

stopWords.add("a");

stopWords.add("able");

stopWords.add("about");

stopWords.add("above");

stopWords.add("abroad");

stopWords.add("according");

stopWords.add("accordingly");

stopWords.add("across");

stopWords.add("actually");

stopWords.add("adj");

stopWords.add("after");

stopWords.add("afterwards");

stopWords.add("again");

stopWords.add("against");

stopWords.add("ago");

stopWords.add("ahead");

stopWords.add("ain");

stopWords.add("all");

stopWords.add("allow");

stopWords.add("allows");

stopWords.add("almost");

stopWords.add("alone");

stopWords.add("along");

stopWords.add("alongside");

stopWords.add("already");

stopWords.add("also");

stopWords.add("although");

stopWords.add("always");

stopWords.add("am");

stopWords.add("amid");

stopWords.add("amidst");

stopWords.add("among");

stopWords.add("amongst");

stopWords.add("an");

stopWords.add("and");

stopWords.add("another");

stopWords.add("any");

stopWords.add("anybody");

stopWords.add("anyhow");

stopWords.add("anyone");

stopWords.add("anything");

stopWords.add("anyway");

stopWords.add("anyways");

stopWords.add("anywhere");

stopWords.add("apart");

stopWords.add("appear");

stopWords.add("appreciate");

stopWords.add("appropriate");

stopWords.add("are");

stopWords.add("aren");

stopWords.add("around");

stopWords.add("as");

stopWords.add("aside");

stopWords.add("ask");

stopWords.add("asking");

stopWords.add("associated");

stopWords.add("at");

stopWords.add("available");

stopWords.add("away");

stopWords.add("awfully");

stopWords.add("b");

stopWords.add("back");

stopWords.add("backward");

stopWords.add("backwards");

stopWords.add("be");

stopWords.add("became");

stopWords.add("because");

stopWords.add("become");

stopWords.add("becomes");

stopWords.add("becoming");

stopWords.add("been");

stopWords.add("before");

stopWords.add("beforehand");

stopWords.add("begin");

stopWords.add("behind");

stopWords.add("being");

stopWords.add("believe");

stopWords.add("below");

stopWords.add("beside");

stopWords.add("besides");

stopWords.add("best");

stopWords.add("better");

stopWords.add("between");

stopWords.add("beyond");

stopWords.add("both");

stopWords.add("brief");

stopWords.add("but");

stopWords.add("by");

stopWords.add("c");

stopWords.add("came");

stopWords.add("can");

stopWords.add("cannot");

stopWords.add("cant");

stopWords.add("caption");

stopWords.add("cause");

stopWords.add("causes");

stopWords.add("certain");

stopWords.add("certainly");

stopWords.add("changes");

stopWords.add("clearly");

stopWords.add("mon");

stopWords.add("co");

stopWords.add("co.");

stopWords.add("com");

stopWords.add("come");

stopWords.add("comes");

stopWords.add("concerning");

stopWords.add("consequently");

stopWords.add("consider");

stopWords.add("considering");

stopWords.add("contain");

stopWords.add("containing");

stopWords.add("contains");

stopWords.add("corresponding");

stopWords.add("could");

stopWords.add("couldn");

stopWords.add("course");

stopWords.add("currently");

stopWords.add("d");

stopWords.add("dare");

stopWords.add("daren");

stopWords.add("definitely");

stopWords.add("described");

stopWords.add("despite");

stopWords.add("did");

stopWords.add("didn");

stopWords.add("different");

stopWords.add("directly");

stopWords.add("do");

stopWords.add("does");

stopWords.add("doesn");

stopWords.add("doing");

stopWords.add("done");

stopWords.add("don");

stopWords.add("down");

stopWords.add("downwards");

stopWords.add("during");

stopWords.add("e");

stopWords.add("each");

stopWords.add("edu");

stopWords.add("eg");

stopWords.add("eight");

stopWords.add("eighty");

stopWords.add("either");

stopWords.add("else");

stopWords.add("elsewhere");

stopWords.add("end");

stopWords.add("ending");

stopWords.add("enough");

stopWords.add("entirely");

stopWords.add("especially");

stopWords.add("et");

stopWords.add("etc");

stopWords.add("even");

stopWords.add("ever");

stopWords.add("evermore");

stopWords.add("every");

stopWords.add("everybody");

stopWords.add("everyone");

stopWords.add("everything");

stopWords.add("everywhere");

stopWords.add("ex");

stopWords.add("exactly");

stopWords.add("example");

stopWords.add("except");

stopWords.add("f");

stopWords.add("fairly");

stopWords.add("far");

stopWords.add("farther");

stopWords.add("few");

stopWords.add("fewer");

stopWords.add("fifth");

stopWords.add("first");

stopWords.add("five");

stopWords.add("followed");

stopWords.add("following");

stopWords.add("follows");

stopWords.add("for");

stopWords.add("forever");

stopWords.add("former");

stopWords.add("formerly");

stopWords.add("forth");

stopWords.add("forward");

stopWords.add("found");

stopWords.add("four");

stopWords.add("from");

stopWords.add("further");

stopWords.add("furthermore");

stopWords.add("g");

stopWords.add("get");

stopWords.add("gets");

stopWords.add("getting");

stopWords.add("given");

stopWords.add("gives");

stopWords.add("go");

stopWords.add("goes");

stopWords.add("going");

stopWords.add("gone");

stopWords.add("got");

stopWords.add("gotten");

stopWords.add("greetings");

stopWords.add("h");

stopWords.add("had");

stopWords.add("hadn");

stopWords.add("half");

stopWords.add("happens");

stopWords.add("hardly");

stopWords.add("has");

stopWords.add("hasn");

stopWords.add("have");

stopWords.add("haven");

stopWords.add("having");

stopWords.add("he");

stopWords.add("ll");

stopWords.add("hello");

stopWords.add("help");

stopWords.add("hence");

stopWords.add("her");

stopWords.add("here");

stopWords.add("hereafter");

stopWords.add("hereby");

stopWords.add("herein");

stopWords.add("hereupon");

stopWords.add("hers");

stopWords.add("herself");

stopWords.add("hi");

stopWords.add("him");

stopWords.add("himself");

stopWords.add("his");

stopWords.add("hither");

stopWords.add("hopefully");

stopWords.add("how");

stopWords.add("howbeit");

stopWords.add("however");

stopWords.add("hundred");

stopWords.add("i");

stopWords.add("ie");

stopWords.add("if");

stopWords.add("ignored");

stopWords.add("immediate");

stopWords.add("in");

stopWords.add("inasmuch");

stopWords.add("inc");

stopWords.add("indeed");

stopWords.add("indicate");

stopWords.add("indicated");

stopWords.add("indicates");

stopWords.add("inner");

stopWords.add("inside");

stopWords.add("insofar");

stopWords.add("instead");

stopWords.add("into");

stopWords.add("inward");

stopWords.add("is");

stopWords.add("isn");

stopWords.add("it");

stopWords.add("its");

stopWords.add("itself");

stopWords.add("ve");

stopWords.add("j");

stopWords.add("just");

stopWords.add("k");

stopWords.add("keep");

stopWords.add("keeps");

stopWords.add("kept");

stopWords.add("know");

stopWords.add("known");

stopWords.add("knows");

stopWords.add("l");

stopWords.add("last");

stopWords.add("lately");

stopWords.add("later");

stopWords.add("latter");

stopWords.add("latterly");

stopWords.add("least");

stopWords.add("less");

stopWords.add("lest");

stopWords.add("let");

stopWords.add("like");

stopWords.add("liked");

stopWords.add("likely");

stopWords.add("likewise");

stopWords.add("little");

stopWords.add("look");

stopWords.add("looking");

stopWords.add("looks");

stopWords.add("low");

stopWords.add("lower");

stopWords.add("ltd");

stopWords.add("m");

stopWords.add("made");

stopWords.add("mainly");

stopWords.add("make");

stopWords.add("makes");

stopWords.add("many");

stopWords.add("may");

stopWords.add("maybe");

stopWords.add("mayn");

stopWords.add("me");

stopWords.add("mean");

stopWords.add("meantime");

stopWords.add("meanwhile");

stopWords.add("merely");

stopWords.add("might");

stopWords.add("mightn");

stopWords.add("mine");

stopWords.add("minus");

stopWords.add("miss");

stopWords.add("more");

stopWords.add("moreover");

stopWords.add("most");

stopWords.add("mostly");

stopWords.add("mr");

stopWords.add("mrs");

stopWords.add("much");

stopWords.add("must");

stopWords.add("mustn");

stopWords.add("my");

stopWords.add("myself");

stopWords.add("n");

stopWords.add("name");

stopWords.add("namely");

stopWords.add("nd");

stopWords.add("near");

stopWords.add("nearly");

stopWords.add("necessary");

stopWords.add("need");

stopWords.add("needn");

stopWords.add("needs");

stopWords.add("neither");

stopWords.add("never");

stopWords.add("neverf");

stopWords.add("neverless");

stopWords.add("nevertheless");

stopWords.add("new");

stopWords.add("next");

stopWords.add("nine");

stopWords.add("ninety");

stopWords.add("no");

stopWords.add("nobody");

stopWords.add("non");

stopWords.add("none");

stopWords.add("nonetheless");

stopWords.add("noone");

stopWords.add("nor");

stopWords.add("normally");

stopWords.add("not");

stopWords.add("nothing");

stopWords.add("notwithstanding");

stopWords.add("novel");

stopWords.add("now");

stopWords.add("nowhere");

stopWords.add("o");

stopWords.add("obviously");

stopWords.add("of");

stopWords.add("off");

stopWords.add("often");

stopWords.add("oh");

stopWords.add("ok");

stopWords.add("okay");

stopWords.add("old");

stopWords.add("on");

stopWords.add("once");

stopWords.add("one");

stopWords.add("ones");

stopWords.add("only");

stopWords.add("onto");

stopWords.add("opposite");

stopWords.add("or");

stopWords.add("other");

stopWords.add("others");

stopWords.add("otherwise");

stopWords.add("ought");

stopWords.add("oughtn");

stopWords.add("our");

stopWords.add("ours");

stopWords.add("ourselves");

stopWords.add("out");

stopWords.add("outside");

stopWords.add("over");

stopWords.add("overall");

stopWords.add("own");

stopWords.add("p");

stopWords.add("particular");

stopWords.add("particularly");

stopWords.add("past");

stopWords.add("per");

stopWords.add("perhaps");

stopWords.add("placed");

stopWords.add("please");

stopWords.add("plus");

stopWords.add("possible");

stopWords.add("presumably");

stopWords.add("probably");

stopWords.add("provided");

stopWords.add("provides");

stopWords.add("q");

stopWords.add("que");

stopWords.add("quite");

stopWords.add("qv");

stopWords.add("r");

stopWords.add("rather");

stopWords.add("rd");

stopWords.add("re");

stopWords.add("really");

stopWords.add("reasonably");

stopWords.add("recent");

stopWords.add("recently");

stopWords.add("regarding");

stopWords.add("regardless");

stopWords.add("regards");

stopWords.add("relatively");

stopWords.add("respectively");

stopWords.add("right");

stopWords.add("round");

stopWords.add("s");

stopWords.add("said");

stopWords.add("same");

stopWords.add("saw");

stopWords.add("say");

stopWords.add("saying");

stopWords.add("says");

stopWords.add("second");

stopWords.add("secondly");

stopWords.add("see");

stopWords.add("seeing");

stopWords.add("seem");

stopWords.add("seemed");

stopWords.add("seeming");

stopWords.add("seems");

stopWords.add("seen");

stopWords.add("self");

stopWords.add("selves");

stopWords.add("sensible");

stopWords.add("sent");

stopWords.add("serious");

stopWords.add("seriously");

stopWords.add("seven");

stopWords.add("several");

stopWords.add("shall");

stopWords.add("shan");

stopWords.add("she");

stopWords.add("should");

stopWords.add("shouldn");

stopWords.add("since");

stopWords.add("six");

stopWords.add("so");

stopWords.add("some");

stopWords.add("somebody");

stopWords.add("someday");

stopWords.add("somehow");

stopWords.add("someone");

stopWords.add("something");

stopWords.add("sometime");

stopWords.add("sometimes");

stopWords.add("somewhat");

stopWords.add("somewhere");

stopWords.add("soon");

stopWords.add("sorry");

stopWords.add("specified");

stopWords.add("specify");

stopWords.add("specifying");

stopWords.add("still");

stopWords.add("sub");

stopWords.add("such");

stopWords.add("sup");

stopWords.add("sure");

stopWords.add("t");

stopWords.add("take");

stopWords.add("taken");

stopWords.add("taking");

stopWords.add("tell");

stopWords.add("tends");

stopWords.add("th");

stopWords.add("than");

stopWords.add("thank");

stopWords.add("thanks");

stopWords.add("thanx");

stopWords.add("that");

stopWords.add("thats");

stopWords.add("ve");

stopWords.add("the");

stopWords.add("their");

stopWords.add("theirs");

stopWords.add("them");

stopWords.add("themselves");

stopWords.add("then");

stopWords.add("thence");

stopWords.add("there");

stopWords.add("thereafter");

stopWords.add("thereby");

stopWords.add("therefore");

stopWords.add("therein");

stopWords.add("theres");

stopWords.add("thereupon");

stopWords.add("these");

stopWords.add("they");

stopWords.add("re");

stopWords.add("thing");

stopWords.add("things");

stopWords.add("think");

stopWords.add("third");

stopWords.add("thirty");

stopWords.add("this");

stopWords.add("thorough");

stopWords.add("thoroughly");

stopWords.add("those");

stopWords.add("though");

stopWords.add("three");

stopWords.add("through");

stopWords.add("throughout");

stopWords.add("thru");

stopWords.add("thus");

stopWords.add("till");

stopWords.add("to");

stopWords.add("together");

stopWords.add("too");

stopWords.add("took");

stopWords.add("toward");

stopWords.add("towards");

stopWords.add("tried");

stopWords.add("tries");

stopWords.add("truly");

stopWords.add("try");

stopWords.add("trying");

stopWords.add("twice");

stopWords.add("two");

stopWords.add("u");

stopWords.add("un");

stopWords.add("under");

stopWords.add("underneath");

stopWords.add("undoing");

stopWords.add("unfortunately");

stopWords.add("unless");

stopWords.add("unlike");

stopWords.add("unlikely");

stopWords.add("until");

stopWords.add("unto");

stopWords.add("up");

stopWords.add("upon");

stopWords.add("upwards");

stopWords.add("us");

stopWords.add("use");

stopWords.add("used");

stopWords.add("useful");

stopWords.add("uses");

stopWords.add("using");

stopWords.add("usually");

stopWords.add("v");

stopWords.add("value");

stopWords.add("various");

stopWords.add("versus");

stopWords.add("very");

stopWords.add("via");

stopWords.add("viz");

stopWords.add("vs");

stopWords.add("w");

stopWords.add("want");

stopWords.add("wants");

stopWords.add("was");

stopWords.add("wasn");

stopWords.add("way");

stopWords.add("we");

stopWords.add("welcome");

stopWords.add("well");

stopWords.add("went");

stopWords.add("were");

stopWords.add("re");

stopWords.add("weren");

stopWords.add("what");

stopWords.add("whatever");

stopWords.add("when");

stopWords.add("whence");

stopWords.add("whenever");

stopWords.add("where");

stopWords.add("whereafter");

stopWords.add("whereas");

stopWords.add("whereby");

stopWords.add("wherein");

stopWords.add("whereupon");

stopWords.add("wherever");

stopWords.add("whether");

stopWords.add("which");

stopWords.add("whichever");

stopWords.add("while");

stopWords.add("whilst");

stopWords.add("whither");

stopWords.add("who");

stopWords.add("whoever");

stopWords.add("whole");

stopWords.add("whom");

stopWords.add("whomever");

stopWords.add("whose");

stopWords.add("why");

stopWords.add("will");

stopWords.add("willing");

stopWords.add("wish");

stopWords.add("with");

stopWords.add("within");

stopWords.add("without");

stopWords.add("wonder");

stopWords.add("won");

stopWords.add("would");

stopWords.add("wouldn");

stopWords.add("x");

stopWords.add("y");

stopWords.add("yes");

stopWords.add("yet");

stopWords.add("you");

stopWords.add("your");

stopWords.add("re");

stopWords.add("yours");

stopWords.add("yourself");

stopWords.add("yourselves");

stopWords.add("z");

stopWords.add("zero");

stopWords.add("de");

stopWords.add("la");

stopWords.add("el");

stopWords.add("@");

stopWords.add("en");

stopWords.add("es");

stopWords.add("te");

stopWords.add("lo");

stopWords.add("mi");

stopWords.add("por");

stopWords.add("con");

stopWords.add("para");

stopWords.add("los");

stopWords.add("1");

stopWords.add("2");

stopWords.add("3");

stopWords.add("4");

stopWords.add("5");

stopWords.add("6");

stopWords.add("7");

stopWords.add("8");

stopWords.add("9");

stopWords.add("0");

stopWords.add("se");

stopWords.add("ya");

stopWords.add("yo");

stopWords.add("si");

}

}

4. IntSumReducer.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public 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);

}

}

5. SortingMapper.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

import java.util.StringTokenizer;

public class SortingMapper extends Mapper<Object, Text, IntWritable, Text> {

private 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());

if (itr.countTokens() == 2) {

word.set(itr.nextToken());

one.set(Integer.parseInt(itr.nextToken()));

context.write(one, word);

}

}

}

6. SortingReducer.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class SortReducer extends Reducer<IntWritable, Text, IntWritable, Text> {

private Text result = new Text();

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

throws IOException, InterruptedException {

for (Text val : values) {

result.clear();

result.set(val);

context.write(key, result);

}

}

}

7. HashTagMapper

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import java.io.IOException;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class HashTagMapper extends Mapper<Object, Text, Text, IntWritable> {

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

private Text word = new Text();

static Pattern p = Pattern.compile("#([^\\s]+)");

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

String temp\_url = value.toString();

String temp = temp\_url.replaceAll("http[s]?://t.co/[A-Za-z0-9]+", " ");

temp = temp.replaceAll("[^A-Za-z0-9@#]", " ");

temp = temp.replaceAll("#+", "#");

Matcher match = p.matcher(temp);

while(match.find()){

word.clear();

word.set(match.group());

context.write(word, one);

}

}

}

8. HashTagReducer

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class HashTagReducer 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);

}

}

9. TweetMapper

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class TweetMapper extends Mapper<Object, Text, Text, IntWritable>{

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

private Text word = new Text();

static Pattern pat = Pattern.compile("@[^\\s]+");

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

String temp\_url = value.toString();

String temp = temp\_url.replaceAll("http[s]?://t.co/[A-Za-z0-9]+", " ");

temp = temp.replaceAll("[^A-Za-z0-9@#]", " ");

temp = temp.replaceAll("#+", "#");

Matcher m = pat.matcher(temp);

while(m.find()){

word.clear();

word.set(m.group());

context.write(word, one);

}

}

}

10. TweetReducer

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package WordCount;

/\*\*

\*

\* @author param

\*/

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class TweetReducer 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);

}

}

TwitterKmeans

1. KMeans.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterkmeans;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

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

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

import java.io.IOException;

import java.util.Map;

public class KMeans {

public static final String INPUT\_PATH = "C:/Users/param//Desktop/hadoop/FinalProject/data/Kmeans/users500.txt";

public static final String OUTPUT\_PATH = "C:/Users/param//Desktop/hadoop/FinalProject/data/Kmeans/output";

public static final int NUM\_CLUSTERS = 50;

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

Configuration conf = new Configuration();

String inputPath = "C:/Users/param//Desktop/hadoop/FinalProject/data/Kmeans/users500.txt";

String outputPath = "C:/Users/param//Desktop/hadoop/FinalProject/data/Kmeans/output";

int K = 10;

boolean isConverged = false;

int maxIterations = 50;

boolean iterationFinished = true;

int numIteration = 0;

Map<Integer, Double> tempCentroids = KMeansUtil.getInitialCentroids(NUM\_CLUSTERS);

KMeansUtil.showOutput(tempCentroids);

while(!isConverged && iterationFinished && (numIteration < maxIterations)) {

numIteration++;

conf.setInt("iteration", numIteration);

KMeansUtil.writeCentroids(conf, tempCentroids);

Job job = Job.getInstance(conf);

job.setJarByClass(KMeans.class);

job.setMapperClass(KMeansMapper.class);

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(LongWritable.class);

job.setReducerClass(KMeansReducer.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(DoubleWritable.class);

FileInputFormat.addInputPath(job, new Path(INPUT\_PATH));

String iterOutputPath = OUTPUT\_PATH + numIteration;

deleteFolder(conf, iterOutputPath);

FileOutputFormat.setOutputPath(job, new Path(iterOutputPath));

iterationFinished = job.waitForCompletion(true);

long convergeCount = job.getCounters().findCounter(KMeansReducer.Convergence.CONVERGENT).getValue();

if(convergeCount == NUM\_CLUSTERS) {

isConverged = true;

}

tempCentroids = KMeansUtil.readCentroids(job);

KMeansUtil.showOutput(tempCentroids);

}

KMeansUtil.showOutput(tempCentroids);

}

private static void deleteFolder(Configuration conf, String folderPath ) throws IOException {

FileSystem fs = FileSystem.get(conf);

Path path = new Path(folderPath);

if(fs.exists(path)) {

fs.delete(path,true);

}

}

}

1. KMeansMapper.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterkmeans;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hdfs.DFSClient;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.MapWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

public class KMeansMapper extends Mapper<Object, Text, IntWritable, LongWritable> {

private static final transient Logger LOG = LoggerFactory.getLogger(KMeansMapper.class);

Map<Integer, Double> centroids = new HashMap<Integer, Double>();

int iteration = -1;

@Override

protected void setup(Context context) throws IOException, InterruptedException {

super.setup(context);

Configuration conf = context.getConfiguration();

iteration = conf.getInt("iteration", -1);

System.out.println("Setting up Mapper " + iteration);

centroids = KMeansUtil.readCentroids(conf);

System.out.println("Mapper " + iteration + " Received " + centroids.size() + " centroids as");

KMeansUtil.showOutput(centroids);

}

@Override

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

LongWritable followerCount = parseEntryToValue(value);

int clusterIndex = -1;

Double minDistance = Double.MAX\_VALUE;

if(followerCount!=null) {

System.out.println("Mapper " + iteration + " map() got value " + value.toString() + " parsed to " + followerCount.get());

for(Integer idx : centroids.keySet()) {

Double distance = Math.abs(centroids.get(idx) - followerCount.get());

if(distance < minDistance) {

minDistance = distance;

clusterIndex = idx;

}

System.out.println("Mapper " + iteration + " Distance of " + centroids.get(idx) + " from " + followerCount.get() + " is " + distance);

}

System.out.println("Mapper " + iteration + " Closest centroid is " + centroids.get(clusterIndex) + " for " + followerCount.get());

context.write(new IntWritable(clusterIndex), followerCount);

} else {

LOG.error("Mapper " + iteration + " follower count received is NULL.");

}

}

LongWritable parseEntryToValue(Text fileEntry) {

LongWritable followerCount = null;

try {

String userEntry = fileEntry.toString();

String parts[] = userEntry.split(",");

Long followerCountValue = Long.parseLong(parts[1].trim());

followerCount = new LongWritable(followerCountValue);

} catch(Exception e) {

followerCount = null;

}

return followerCount;

}

}

1. KMeansReducer.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterkmeans;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Reducer;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

public class KMeansReducer extends Reducer<IntWritable, LongWritable, IntWritable, DoubleWritable> {

private static final transient Logger LOG = LoggerFactory.getLogger(KMeansReducer.class);

int iteration = -1;

Map<Integer, Double> centroids = new HashMap<Integer, Double>();

public static enum Convergence {

DIVERGENT, CONVERGENT

}

@Override

protected void setup(Reducer.Context context) throws IOException, InterruptedException {

super.setup(context);

Configuration conf = context.getConfiguration();

iteration = conf.getInt("iteration", -1);

System.out.println("Setting up Reducer " + iteration);

centroids = KMeansUtil.readCentroids(conf);

System.out.println("Reducer " + iteration + " Received " + centroids.size() + " centroids as");

KMeansUtil.showOutput(centroids);

}

@Override

protected void reduce(IntWritable key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException {

Double currentCentroid = centroids.get(key.get());

System.out.println("Reducer " + iteration + " Current Centroid: " + currentCentroid);

Double newCentroid = getNewCentroid(values);

System.out.println("Reducer " + iteration + " Reducer for centroid " + key.get() + " which is " + centroids.get(key.get()));

if(newCentroid != null) {

System.out.println("Reducer " + iteration + " New centroid calculated for " + key.get() + " is " + newCentroid);

if(belowThreshold(currentCentroid, newCentroid)) {

context.getCounter(Convergence.CONVERGENT).increment(1);

System.out.println("Reducer " + iteration + " REDUCER CONVERGED!" + context.getCounter(Convergence.CONVERGENT).getValue() + " times");

}

context = KMeansUtil.writeCentroid(context, key.get(), newCentroid);

DoubleWritable op = new DoubleWritable(newCentroid);

context.write(key, op);

} else {

System.out.println("Reducer " + iteration + " New Centroid calculated is NULL!");

}

}

private boolean belowThreshold(Double currentCentroid, Double newCentroid) {

return Math.abs(currentCentroid - newCentroid) <= KMeansUtil.THRESHOLD;

}

private Double getNewCentroid(Iterable<LongWritable> values) {

Double average = null;

int numValues = 0;

long total = 0L;

for(LongWritable val : values) {

total += val.get();

numValues++;

}

if(numValues > 0) {

average = (double) (total/numValues);

}

return average;

}

}

1. KMeansUtil.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterkmeans;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

import java.math.RoundingMode;

import java.text.DecimalFormat;

import java.util.HashMap;

import java.util.Map;

public class KMeansUtil {

public static final double THRESHOLD = 0.005;

public static final String KEY\_COUNTER = "CENTROID\_";

public static final String KEY\_GROUP = "KMEANS";

public static final String CONF\_NUM\_CENTROIDS = "iteration.centroids.count";

public static final String CONF\_CENTROID\_KEY = "iteration.centroid.";

public static Map<Integer, Double> getInitialCentroids(int numClusters) {

Map<Integer, Double> initCentroids = new HashMap<Integer, Double>();

int i = 1;

long v = 0;

while(i <= numClusters) {

initCentroids.put(i, (double)v);

v += 1000\*(++i);

}

return initCentroids;

}

public static Map<Integer, Double> readCentroids(Job job) throws IOException {

Map<Integer, Double> iterCentroids = new HashMap<Integer, Double>();

int num = job.getConfiguration().getInt(CONF\_NUM\_CENTROIDS, -1);

int i = 1;

while(i <= num) {

long value = job.getCounters().findCounter(KEY\_GROUP, KEY\_COUNTER + i).getValue();

double dblValue = value/1000;

iterCentroids.put(i, dblValue);

i++;

}

return iterCentroids;

}

public static Reducer.Context writeCentroid(Reducer.Context context, int key, Double value) {

long undecimalVal = undecimal(value);

System.out.println("KMeansUtil: Modifying counter in Reducer for centroid " + key + " from " +

context.getCounter(KEY\_GROUP, KEY\_COUNTER + key).getValue() +

" to " + value + " as " + undecimalVal);

context.getCounter(KEY\_GROUP, KEY\_COUNTER + key).setValue(undecimalVal);

return context;

}

private static long undecimal(double decimalVal) {

long undecimalVal = -1;

try {

DecimalFormat df = new DecimalFormat("#.###");

df.setMaximumFractionDigits(3);

df.setRoundingMode(RoundingMode.UP);

String formattedVal = df.format(decimalVal);

double formattedDecimalVal = Double.valueOf(formattedVal);

undecimalVal = (long)(formattedDecimalVal \* 1000);

} catch (Exception e) {

undecimalVal = -1;

}

return undecimalVal;

}

public static void writeCentroids(Configuration conf, Map<Integer, Double> tempCentroids) {

int num = tempCentroids.size();

conf.setInt(CONF\_NUM\_CENTROIDS, num);

int i = 1;

for(Integer entry : tempCentroids.keySet()) {

double decimalVal = tempCentroids.get(entry);

conf.setDouble(CONF\_CENTROID\_KEY + i, decimalVal);

i++;

}

}

public static Map<Integer, Double> readCentroids(Configuration conf) {

Map<Integer, Double> iterCentroids = new HashMap<Integer, Double>();

int num = conf.getInt(CONF\_NUM\_CENTROIDS, -1);

System.out.println("KMeansUtil: Reading " + num + " centroids from Configuration");

boolean isNeg = false;

if(num > 0) {

int i = 1;

while(i <= num) {

double centroid = conf.getDouble(CONF\_CENTROID\_KEY + i, -1.0);

if(centroid >= 0) {

System.out.println("KMeansUtil: Found Centroid " + i + " : " + centroid + " in Configuration");

iterCentroids.put(i, centroid);

i++;

isNeg = false;

} else {

if(!isNeg) {

System.out.println("KMeansUtil: Centroid not found for " + CONF\_CENTROID\_KEY + i);

isNeg = true;

} else {

System.out.print("."+i);

}

i++;

}

}

}

return iterCentroids;

}

public static void showOutput(Map<Integer, Double> tempCentroids) {

for(Map.Entry<Integer, Double> entry : tempCentroids.entrySet()) {

System.out.println(entry.getKey() + " : " + entry.getValue());

}

}

}

TwitterWordCooccurence

1. WordCount.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterwordcooccurence;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

import java.io.IOException;

public class WordCount {

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

Configuration conf = new Configuration();

String inputPathWordCount = "C:/Users/param//Desktop/hadoop/FinalProject/data/tweets.txt";

String outputPathWordCount = "C:/Users/param//Desktop/hadoop/FinalProject/data/output";

String outputPathPair = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/pair";

String outputPathStripes = "C:/Users/param//Desktop/hadoop/FinalProject/data/output/stripes";

deleteFolder(conf, outputPathWordCount);

deleteFolder(conf, outputPathPair);

deleteFolder(conf, outputPathStripes);

deleteFolder(conf, outputPathWordCount);

Job job = Job.getInstance(conf);

Job job2 = null;

job.setJarByClass(WordCount.class);

job.setMapperClass(PairMapper.class);

job.setReducerClass(PairReducer.class);

job.setNumReduceTasks(2);

job.setPartitionerClass(PairPartitioner.class);

job.setSortComparatorClass(PairComparator.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(DoubleWritable.class);

FileInputFormat.addInputPath(job, new Path(inputPathWordCount));

FileOutputFormat.setOutputPath(job, new Path(outputPathPair));

if (job.waitForCompletion(true)) {

job2 = Job.getInstance(conf);

job2.setJarByClass(WordCount.class);

job2.setMapperClass(StripeMapper.class);

job2.setCombinerClass(StripeReducer.class);

job2.setReducerClass(StripeReducer.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job2, new Path(inputPathWordCount));

FileOutputFormat.setOutputPath(job2, new Path(outputPathStripes));

System.exit(job2.waitForCompletion(true) ? 0 : 1);

}

}

private static void deleteFolder(Configuration conf, String folderPath)

throws IOException {

FileSystem fs = FileSystem.get(conf);

Path path = new Path(folderPath);

if (fs.exists(path)) {

fs.delete(path, true);

}

}

}

1. StopWords.java
2. PairMapper.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterwordcooccurence;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashSet;

import java.util.StringTokenizer;

public class PairMapper extends Mapper<Object, Text, Text, DoubleWritable> {

private final static DoubleWritable one = new DoubleWritable(1);

private Text word = new Text();

private DoubleWritable sumValue = new DoubleWritable(1);

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

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

String temp\_url = value.toString();

String temp = temp\_url.replaceAll("http[s]?://t.co/[A-Za-z0-9]+", " ");

temp = temp.replaceAll("[^A-Za-z0-9@#]", " ");

temp = temp.replaceAll("#+", "#");

HashSet<String> set = new HashSet<String>();

StringTokenizer itr = new StringTokenizer(temp);

while (itr.hasMoreTokens()) {

String tempStr = itr.nextToken();

if (!StopWords.stopWords.contains(tempStr.toLowerCase())) {

words.add(tempStr);

}

}

Collections.sort(words);

for (int i = 0; i < words.size(); i++) {

int sum = 0;

if (set.contains(words.get(i)) || "RT".equals(words.get(i))) {

continue;

} else {

set.add(words.get(i));

for (int j = 0; j < words.size(); j++) {

if (i == j || "RT".equals(words.get(j))) {

continue;

}

sum += 1;

word.clear();

word.set(words.get(i) + "," + words.get(j));

context.write(word, one);

}

word.clear();

word.set(words.get(i) + ",\*");

sumValue.set(sum);

context.write(word, sumValue);

}

}

}

}

1. PairComparator.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterwordcooccurence;

/\*\*

\*

\* @author param

\*/

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

public class PairComparator extends WritableComparator implements

WritableComparable<Text> {

protected PairComparator() {

super(Text.class, true);

}

@Override

public void readFields(DataInput arg0) throws IOException {

// TODO Auto-generated method stub

}

@Override

public void write(DataOutput arg0) throws IOException {

// TODO Auto-generated method stub

}

@Override

public int compareTo(Text o) {

String data1[] = this.toString().split(",");

String data2[] = o.toString().split(",");

int compareValue = data1[0].compareTo(data2[0]);

if (compareValue == 0) {

if (data1[1].equals("\*") && data2[1].equals("\*")) {

return 0;

} else if (data1[1].equals("\*")) {

return -1;

} else if (data2[1].equals("\*")) {

return 1;

} else {

return data1[1].compareTo(data2[1]);

}

} else {

return compareValue;

}

}

@SuppressWarnings("rawtypes")

@Override

public int compare(WritableComparable w1, WritableComparable w2) {

Text t1 = (Text) w1;

Text t2 = (Text) w2;

String data1[] = t1.toString().split(",");

String data2[] = t2.toString().split(",");

int compareValue = data1[0].compareTo(data2[0]);

if (compareValue == 0) {

if (data1[1].equals("\*") && data2[1].equals("\*")) {

return 0;

} else if (data1[1].equals("\*")) {

return -1;

} else if (data2[1].equals("\*")) {

return 1;

} else {

return data1[1].compareTo(data2[1]);

}

} else {

return compareValue;

}

}

}

1. PairReducer.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterwordcooccurence;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class PairReducer extends

Reducer<Text, DoubleWritable, Text, DoubleWritable> {

private DoubleWritable result = new DoubleWritable();

private DoubleWritable val = new DoubleWritable();

@Override

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

Context context) throws IOException, InterruptedException {

double sum = 0;

double finalSum = 0;

for (DoubleWritable val : values) {

sum += val.get();

}

String temp = key.toString().split(",")[1];

if ("\*".equals(temp)) {

val.set(sum);

result.set(sum);

} else {

finalSum = (sum / (val.get()));

result.set(finalSum);

}

context.write(key, result);

}

}

1. StripeMapper.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterwordcooccurence;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

import java.util.\*;

public class StripeMapper extends Mapper<Object, Text, Text, Text> {

private HashMap<String, Double> map = new HashMap<String, Double>();

private Text word = new Text();

private Text text = new Text();

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

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

String temp\_url = value.toString();

HashSet<String> set = new HashSet<String>();

String temp = temp\_url.replaceAll("http[s]?://t.co/[A-Za-z0-9]+", " ");

temp = temp.replaceAll("[^A-Za-z0-9@#]", " ");

temp = temp.replaceAll("#+", "#");

StringTokenizer itr = new StringTokenizer(temp);

while (itr.hasMoreTokens()) {

String tempStr = itr.nextToken();

if (!StopWords.stopWords.contains(tempStr.toLowerCase())) {

words.add(tempStr);

}

}

Collections.sort(words);

for (int i = 0; i < words.size(); i++) {

map.clear();

word.clear();

word.set(words.get(i));

if (set.contains(words.get(i)) || "RT".equals(words.get(i))) {

continue;

} else {

set.add(words.get(i));

for (int j = 0; j < words.size(); j++) {

String m = null;

if (i != j && (!"RT".equals(words.get(j)))) {

m = words.get(j);

if (map.containsKey(m)) {

double data = map.get(m) + 1;

map.put(m, data);

} else {

map.put(m, 1d);

}

} else {

continue;

}

}

text.clear();

text.set(getMapData(map));

context.write(word, text);

}

}

}

public String getMapData(HashMap<String, Double> m) {

if (m != null && m.size() > 0) {

String returnString = "";

Set<String> keySet = m.keySet();

Iterator<String> it = keySet.iterator();

while (it.hasNext()) {

String temp = it.next();

double value = m.get(temp);

returnString += temp + "," + value + ";";

}

return returnString;

}

return "";

}

}

1. StripeReducer.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twitterwordcooccurence;

/\*\*

\*

\* @author param

\*/

import java.io.IOException;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Set;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class StripeReducer extends Reducer<Text, Text, Text, Text> {

private Text text = new Text();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

HashMap<String, Double> finalMap = new HashMap<String, Double>();

double sum = 0;

for (Text val : values) {

String data[] = val.toString().split(";");

for (int i = 0; i < data.length; i++) {

if (!"".equals(data[i])) {

String newData[] = data[i].split(",");

sum += Double.parseDouble(newData[1]);

if (finalMap.containsKey(newData[0])) {

double d = finalMap.get(newData[0])

+ Double.parseDouble(newData[1]);

finalMap.put(newData[0], d);

} else {

finalMap.put(newData[0], Double.parseDouble(newData[1]));

}

}

}

}

text.clear();

text.set(getMapData(finalMap, sum));

context.write(key, text);

}

public String getMapData(HashMap<String, Double> m, double sum) {

if (m != null && m.size() > 0) {

String returnString = "";

Set<String> keySet = m.keySet();

Iterator<String> it = keySet.iterator();

while (it.hasNext()) {

String temp = it.next();

double value = (m.get(temp) / sum);

returnString += temp + "," + value + ";";

}

return returnString;

}

return "";

}

}

TwitterShortestSocialGraph

1. ShortestPath.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twittershortestpathsocialgraph;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

import java.io.IOException;

public class ShortestPath {

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

Configuration conf = new Configuration();

String inputPathSocialGraph = "C:/Users/param//Desktop/hadoop/FinalProject/data/SocialGraph/input-graph-large";

String outputPathSocialGraph = "C:/Users/param//Desktop/hadoop/FinalProject/data/SocialGraph/output";

deleteFolder(conf, outputPathSocialGraph);

boolean isCompleted = false;

int count = 0;

while (isCompleted == false) {

String input = null;

String output = null;

if (count == 0) {

input = inputPathSocialGraph;

} else {

input = outputPathSocialGraph + "/" + count;

}

count++;

output = outputPathSocialGraph + "/" + count;

deleteFolder(conf, output);

Job job = Job.getInstance(conf);

job.setJarByClass(ShortestPath.class);

job.setMapperClass(NodeMapper.class);

job.setReducerClass(NodeReducer.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output));

boolean isJobCompleted = job.waitForCompletion(true);

if (isJobCompleted) {

isCompleted = (job.getCounters().getGroup("group").findCounter("isModified").getValue() > 0 ? false : true);

}

}

}

private static void deleteFolder(Configuration conf, String folderPath)

throws IOException {

FileSystem fs = FileSystem.get(conf);

Path path = new Path(folderPath);

if (fs.exists(path)) {

fs.delete(path, true);

}

}

}

1. NodeMapper.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twittershortestpathsocialgraph;

/\*\*

\*

\* @author param

\*/

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class NodeMapper extends Mapper<Object, Text, LongWritable, Text> {

private Text word = new Text();

private LongWritable lo = new LongWritable();

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

if (!"".equals(value.toString())) {

String data[] = value.toString().split("[ \t]+");

long nodeId = Long.parseLong(data[0]);

long distance = Long.parseLong(data[1]);

String adjacencyList[] = data[2].split(":");

word.clear();

word.set("node," + data[2]);

lo.set(nodeId);

context.write(lo, word);

word.clear();

word.set("origin\_distance," + distance);

context.write(lo, word);

distance++;

word.clear();

word.set("distance," + distance);

for (int i = 0; i < adjacencyList.length; i++) {

lo.set(Long.parseLong(adjacencyList[i]));

context.write(lo, word);

}

}

}

}

1. NodeReducer.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package twittershortestpathsocialgraph;

/\*\*

\*

\* @author param

\*/

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class NodeReducer extends

Reducer<LongWritable, Text, LongWritable, Text> {

Text word = new Text();

@Override

public void reduce(LongWritable key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

long lowest = -1;

long originDistance = -1;

String node = null;

for (Text t : values) {

String data[] = t.toString().split(",");

if (null != data[0])

switch (data[0]) {

case "node":

node = data[1];

break;

case "distance":

long distance = Long.parseLong(data[1]);

if (lowest == -1) {

lowest = distance;

} else {

if (lowest > distance) {

lowest = distance;

}

}

break;

case "origin\_distance":

originDistance = Long.parseLong(data[1]);

break;

default:

break;

}

}

if (originDistance > lowest && lowest > -1){

context.getCounter("group", "isModified").increment(1);

} else {

lowest = originDistance;

}

word.clear();

word.set(lowest + " " + node);

context.write(key, word);

}

}