Hw3-1: Hadoop

15826 – Multimedia Databases and Data Mining

Fall 2013, C. Faloutsos

Emma R. Zhang{runyunz@andrew.cmu.edu} - November 18, 2013

Code

NGram.java

```
//package org.myorg;
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class NGram extends Configured implements Tool{

     public static class Map extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
          private final static IntWritable one = new IntWritable(1);
          private Text word = new Text();
          private int n = 0;

          public void configure(JobConf job) {
               n = Integer.parseInt(job.get("ngram.n"));
               // m = Integer.parseInt(job.get("ngram.m"));
          }

          public void map(LongWritable key, Text value,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException {
               String line = value.toString();
               StringTokenizer tokenizer = new
StringTokenizer(line);

               List<String> sList = new ArrayList<String>();
```

```
                StringBuffer sb = new StringBuffer();

                if (tokenizer.countTokens() >= n) {
                     for (int i = 0; i < n; ++i) {
                          sList.add(tokenizer.nextToken());
                          sb.append(sList.get(i));
                          sb.append("+");
                     }
                     word.set(sb.substring(0,sb.length()-1));
                     output.collect(word, one);

                     while (tokenizer.hasMoreTokens()) {
                          sb.delete(0, sb.length());

                          for(int i = 0; i < n-1; ++i){
                               sList.set(i, sList.get(i+1));
                               sb.append(sList.get(i));
                               sb.append("+");
                          }

                          sList.set(sList.size()-1,
tokenizer.nextToken());
                          sb.append(sList.get(sList.size()-1));

                          word.set(sb.toString());
                          output.collect(word, one);
                     }
                }
           }
      }

      public static class Reduce extends MapReduceBase implements
Reducer<Text, IntWritable, Text, IntWritable> {
           private int m = 0;

           public void configure(JobConf job) {
                // n = Integer.parseInt(job.get("ngram.n"));
                m = Integer.parseInt(job.get("ngram.m"));
           }

           public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException {
                int sum = 0;
                while (values.hasNext()) {
                     sum += values.next().get();
```

```
                }
                if (sum >= m) {
                    output.collect(key, new IntWritable(sum));
                }
            }
        }

        public int run(String[] args) throws Exception {
            JobConf conf = new JobConf(getConf(), NGram.class);
            conf.setJobName("ngram");

            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);

            conf.setMapperClass(Map.class);
            //conf.setCombinerClass(Reduce.class);
            conf.setReducerClass(Reduce.class);

            conf.setInputFormat(TextInputFormat.class);
            conf.setOutputFormat(TextOutputFormat.class);

            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));

            //conf.set("ngram.n", "2");
            //conf.set("ngram.m", "100");

            JobClient.runJob(conf);
            return 0;
        }

        public static void main(String[] args) throws Exception {
            int res = ToolRunner.run(new Configuration(), new NGram(),
args);
            System.exit(res);
        }
}
```

q1.sh

```
echo "Q0: Clean up directory"
bin/hadoop fs -rmr /reddit_titles/
bin/hadoop fs -rmr /reddit_output/
echo "----------------------------------------"
echo "Q1: Create directory and upload dataset"
bin/hadoop fs -mkdir /reddit_titles/
bin/hadoop fs -put reddit_titles/reddit_titles.csv /reddit_titles/
echo "----------------------------------------"
echo "Q2: Compile WordCount"
cd ngram
./compile.sh WordCount
cd ..
echo "----------------------------------------"
echo "Q3: Run WordCount and report the number of unique words"
bin/hadoop jar ngram/WordCount.jar WordCount /reddit_titles/ /
reddit_output/word_count
echo "Number unique words"
bin/hadoop fs -cat /reddit_output/word_count/part-00000 | wc -l
echo "----------------------------------------"
echo "Q4: Run NGram and Output"
cd ngram
./compile.sh NGram
cd ..
echo "----------------------------------------"
echo "Q5: Report number of bigrams(n=2) occur at least 100 times"
bin/hadoop jar ngram/NGram.jar NGram -Dngram.n=2 -Dngram.m=100 /
reddit_titles/ /reddit_output/ngram
echo "Number of trigrams(n=2) occur at least 100 times"
bin/hadoop fs -cat /reddit_output/ngram/part-00000 | wc -l
echo "----------------------------------------"
echo "Q6: List the top 20 most common bigrams and the number of
times they occur"
bin/hadoop fs -cat /reddit_output/ngram/part-00000 | sort -nrk 2 |
head -n20
echo "----------------------------------------"
echo "Q7: Report number of trigrams(n=3) occur at least 20 times"
bin/hadoop jar ngram/NGram.jar NGram -Dngram.n=3 -Dngram.m=20 /
reddit_titles/ /reddit_output/tringram
echo "Number of trigrams(n=3) occur at least 20 times"
bin/hadoop fs -cat /reddit_output/tringram/part-00000 | wc -l
echo "----------------------------------------"
echo "Q8: List the top 20 most common trigrams and the number of
times they occur"
```

```
bin/hadoop fs -cat /reddit_output/tringram/part-00000 | sort -nrk 2
| head -n20
```

compile.sh

```
#!/bin/bash

function buildJar {
        rm -rf $1_classes/
        mkdir $1_classes
        javac -classpath ../hadoop-core-1.2.1.jar -d $1_classes/
$1.java
        jar -cf $1.jar -C $1_classes/ .
        rm -rf $1_classes/
}

buildJar $1
# buildJar WordCount
# buildJar NGram
```

Result

Q0: Clean up directory
Deleted hdfs://localhost:9000/reddit_titles
Deleted hdfs://localhost:9000/reddit_output

Q1: Create directory and upload dataset

Q2: Compile WordCount

Q3: Run WordCount and report the number of unique words
Number unique words
38064

Q4: Run NGram and Output

Q5: Report number of bigrams(n=2) occur at least 100 times
Number of trigrams(n=2) occur at least 100 times
599

Q6: List the top 20 most common bigrams and the number of times they occur
how+i 5606
i+feel 4230
when+i 3692
this+is 3405
in+the 2149
of+the 1981
xpost+from 1614
on+the 1614
i+see 1369
feel+when 1307
in+a 1214
i+dont 1192
to+the 1147
every+time 1138
i+think 1080
on+my 1066
when+my 990
for+the 984
i+have 964
what+i 939

Q7: Report number of trigrams(n=3) occur at least 100 times
Number of trigrams(n=3) occur at least 100 times
1533

Q8: List the top 20 most common trigrams and the number of times they occur
how+i+feel 3904
i+feel+when 1259
this+is+how 790
every+time+i679
the+front+page 616
how+i+felt 595
feel+when+i 581
is+how+i 548
my+cake+day 482
this+is+what 473
when+i+see 404
i+feel+after 395
the+first+time 390
i+feel+about 361
my+reaction+when 315
i+see+a 312
one+of+my 300
for+the+first 299
xpost+from+rfunny 298
this+is+the 285