FINAL PROJECT

Kevin Dang, Ethan Higa, Kevin Wolff

APRIL 26, 2023

GONZAGA UNIVERSITY

Contents

Introduction

For this project, the input is a text file containing words and the output is the confidence of two specific words. Confidence can be defined as $\frac{(x,y)}{x} = z$ where x and y are two different objects, the denominator is the number of occurrences of the two objects together, and the numerator is the number of occurrences of one of the objects. It can be read as the confidence of x appearing with y is z. For this program, the numerator is the number of lines having a unique combination of two words and the denominator is the number of lines with an individual word from the combination. Hadoop was used for this project to simplify parallelization. It does this by dividing parallelization into 2 main steps: mapping data and reducing data. The mapping function takes an input and outputs key value pairs into an intermediate result that the reducing function reduces to unique keys. The details of how the input is mapped/ reduced is up to the developer, but the specifics of items like load balancing and master/ worker nodes are abstracted by the software.

Performance Analysis

From increasing the amount of Shakespeare text files, the first stage with the term pair count map-reduce takes more time but the time for the second stage for the confidence map-reduce does not change. This is likely due to the implications of memory bottlenecks from writing and reading.

How many Complete_Shakespeare.txt	Runtime (ms)	Added up times (ms)
1	22175	142+10760+143+11130
2	40072	136+11380+656+27900
3	54838	166+12310+922+41440

Discussion and Conclusion

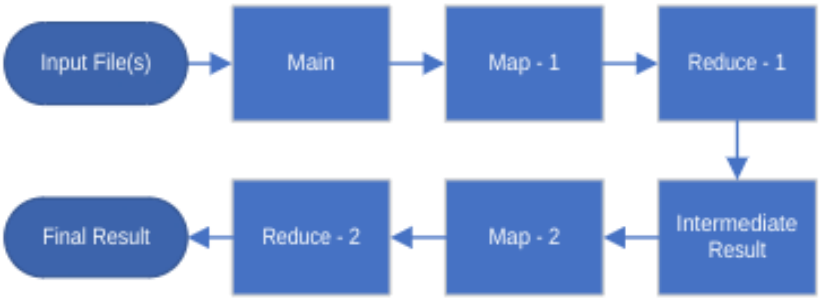
The first map-reduce outputs to an intermediate file which is then inputted into the second map-reduce which creates the final output. Writing and reading from a file is the main bottleneck and takes significantly more time than computations and parsing within the program thus the addition of more input files increases the program time to completion in a non-linear way. This is because adding input files does not increase the number of intermediate files or output files and only primarily increases the time of completion for the first map function. The addition of input files may significantly increase the computational load, but that computational load is relatively small compared to the total write and read time.

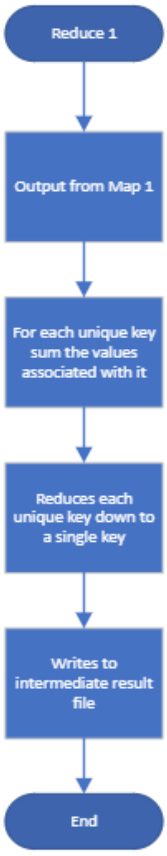
Contributions

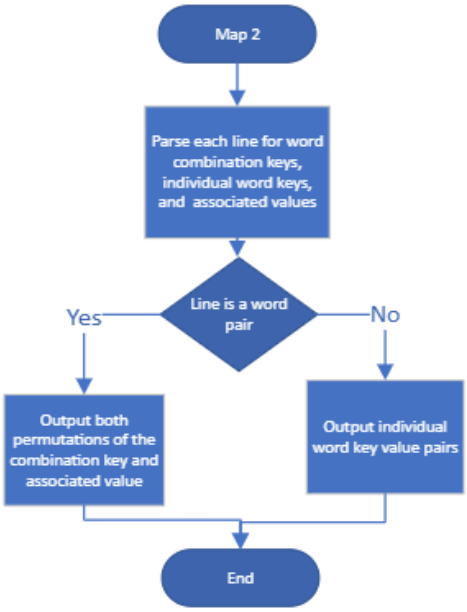
We all coded and white boarded.

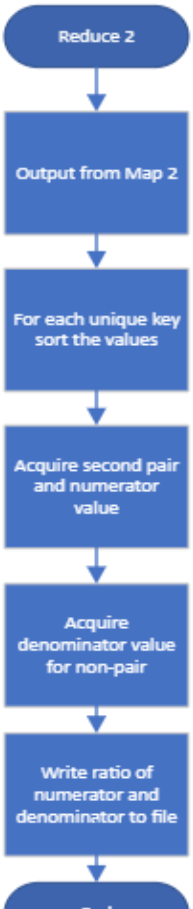
Implementation

This is our implementation of MapReduce:









Appendix

Successful Compilation and Execution

```
2023-04-27 01:43:02,181 INFO mapreduce.Job: Counters: 54
File System Counters
FILE: Number of bytes read=19097762
FILE: Number of bytes written=38741351
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=8591121
HDFS: Number of bytes written=33923626
HDFS: Number of read operations=8
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
HDFS: Number of bytes read erasure-coded=0
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=4781
Total time spent by all reduces in occupied slots (ms)=6106
Total time spent by all map tasks (ms)=4781
Total time spent by all reduce tasks (ms)=6106
Total vcore-milliseconds taken by all map tasks=4781
Total vcore-milliseconds taken by all reduce tasks=6106
Total megabyte-milliseconds taken by all map tasks=4895744
Total megabyte-milliseconds taken by all reduce tasks=6252544
Map-Reduce Framework
Map input records=573100
Map output records=1116628
Map output bytes=16864500
Map output materialized bytes=19097762
Input split bytes=133
Combine input records=0
Combine output records=0
Reduce input groups=29572
Reduce shuffle bytes=19097762
Reduce input records=1116628
Reduce output records=1087056
Spilled Records=2233256
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=142
CPU time spent (ms)=10760
Physical memory (bytes) snapshot=792973312
Virtual memory (bytes) snapshot=5094297600
Total committed heap usage (bytes)=642252800
Peak Map Physical memory (bytes)=420622336
Peak Map Virtual memory (bytes)=2541772800
Peak Reduce Physical memory (bytes)=372350976
Peak Reduce Virtual memory (bytes)=2552524800
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=8590988
```

```
File Output Format Counters
Bytes Written=33923626
```

Output

From file.txt

```
Hello to World bye World!
Hello kevin# Bye Kevin
```

The output is:

```
ehiga@ece-hadoop-01:~/CPEN435_KKE_Project/confidence$ ./cat.sh
2023-04-27 01:07:35,232 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
bye:hello    1.0
bye:kevin    0.5
bye:world    0.5
hello:bye    1.0
hello:kevin  0.5
hello:world  0.5
kevin:bye    1.0
kevin:hello  1.0
world:bye    1.0
world:hello  1.0
```

From Complete_Shakespeare.txt the output is

```
zanies:better1.0
zanies:fools  1.0
zanies:kind   1.0
zanies:set    1.0
zany:carry-tale     1.0
zany:please-man     1.0
zany:slight   1.0
zeal:accuse   0.030303030303030304
zeal:ardent   0.030303030303030304
zeal:as--in   0.030303030303030304
zeal:bardolph0.030303030303030304
zeal:begin    0.030303030303030304
zeal:better   0.030303030303030304
zeal:bore     0.030303030303030304
zeal:boy      0.030303030303030304
zeal:breath   0.030303030303030304
zeal:bright   0.030303030303030304
zeal:brought  0.030303030303030304
zeal:buckingham     0.030303030303030304
zeal:burns    0.030303030303030304
zeal:charity  0.030303030303030304
zeal:christian      0.030303030303030304
zeal:claudio  0.030303030303030304
zeal:cold     0.06060606060606061
zeal:constraint     0.030303030303030304
zeal:contents0.030303030303030304
zeal:content  0.030303030303030304
zeal:cool     0.030303030303030304
zeal:counterfeited  0.030303030303030304
zeal:crackd   0.030303030303030304
zeal:create   0.030303030303030304
zeal:dear     0.030303030303030304
zeal:deep     0.030303030303030304
```

```
zeal:devotion 0.06060606060606061
zeal:dies     0.030303030303030304
zeal:doth     0.030303030303030304
zeal:duty     0.06060606060606061
zeal:faith    0.06060606060606061
zeal:field    0.030303030303030304
zeal:fire     0.030303030303030304
zeal:freeze   0.030303030303030304
zeal:fury     0.030303030303030304
zeal:god      0.06060606060606061
zeal:grace    0.06060606060606061
zeal:half     0.030303030303030304
zeal:hath     0.030303030303030304
zeal:hearts   0.030303030303030304
zeal:he       0.06060606060606061
zeal:highly   0.030303030303030304
zeal:him      0.030303030303030304
zeal:his      0.06060606060606061
zeal:honest   0.030303030303030304
zeal:hospitable     0.030303030303030304
zeal:infer    0.030303030303030304
zeal:infringed      0.030303030303030304
zeal:innocency      0.030303030303030304
zeal:inspird  0.030303030303030304
zeal:integrity      0.030303030303030304
zeal:it-as    0.030303030303030304
zeal:kind     0.030303030303030304
zeal:king     0.030303030303030304
zeal:league   0.030303030303030304
zeal:lest     0.030303030303030304
zeal:love     0.06060606060606061
zeal:melted   0.030303030303030304
zeal:methinks 0.030303030303030304
zeal:mind     0.030303030303030304
zeal:modesty  0.030303030303030304
zeal:movd     0.030303030303030304
zeal:mowbray  0.030303030303030304
zeal:my       0.12121212121212122
zeal:obedience      0.030303030303030304
zeal:on       0.030303030303030304
zeal:ours     0.030303030303030304
zeal:people   0.030303030303030304
zeal:petition 0.030303030303030304
zeal:presents 0.030303030303030304
zeal:prevails 0.030303030303030304
zeal:prince   0.030303030303030304
zeal:quench   0.030303030303030304
zeal:realms   0.030303030303030304
zeal:refuse   0.030303030303030304
zeal:right    0.06060606060606061
zeal:sect     0.030303030303030304
zeal:servd    0.030303030303030304
zeal:set      0.030303030303030304
zeal:shame    0.030303030303030304
zeal:she      0.030303030303030304
zeal:strives  0.030303030303030304
zeal:swear    0.030303030303030304
zeal:tears    0.030303030303030304
zeal:terms    0.06060606060606061
zeal:thee     0.030303030303030304
zeal:thy      0.030303030303030304
zeal:true     0.030303030303030304
zeal:twill    0.030303030303030304
```

```
zeal:under    0.030303030303030304
zeal:unmatched      0.030303030303030304
zeal:unto     0.030303030303030304
zeal:unurgd   0.030303030303030304
zeal:upright  0.030303030303030304
zeal:valentine      0.030303030303030304
zeal:voluntary      0.030303030303030304
zeal:wanted   0.030303030303030304
```

Code

```
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.Collections;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import java.io.FileReader;
import java.net.URI;

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;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.util.StringUtils;

public class Confidence {

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

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    // for caseSensitive
    private boolean caseSensitive;
    private Set<String> patternsToSkip = new HashSet<String>();
    private Set<String> stopWordsToSkip = new HashSet<String>();
    private Configuration conf;
    private BufferedReader fis;

    @Override
    public void setup(Context context) throws IOException,
             InterruptedException {
       conf = context.getConfiguration();
       caseSensitive = conf.getBoolean("confidence.case.sensitive", true);
       //not case sensitive
       if (conf.getBoolean("confidence.skip.patterns", false)) {
              URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
              for (URI patternsURI : patternsURIs) {
                     Path patternsPath = new Path(patternsURI.getPath());
                     String patternsFileName = patternsPath.getName().toString();
                         if(patternsFileName.equals("patterns.txt")) {
                     parseSkipFile(patternsFileName); }
                     if(patternsFileName.equals("stop_words.txt")) {
                     parseStopWordsFile(patternsFileName); }
              }
       }
    }

    private void parseSkipFile(String fileName) {
```

```
       try {
              fis = new BufferedReader(new FileReader(fileName));
              String pattern = null;
              while ((pattern = fis.readLine()) != null) {
                     patternsToSkip.add(pattern);
              }
       } catch (IOException ioe) {
              System.err.println("Caught exception while parsing the cached file '"
              + StringUtils.stringifyException(ioe));
       }
     }

    private void parseStopWordsFile(String fileName) {
       try {
              fis = new BufferedReader(new FileReader(fileName));
              String pattern = null;
              while ((pattern = fis.readLine()) != null) {
                     stopWordsToSkip.add(pattern);
              }
       } catch (IOException ioe) {
              System.err.println("Caught exception while parsing the cached file '"
              + StringUtils.stringifyException(ioe));
       }
     }



    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      ArrayList<String> A = new ArrayList<String>(); // String not Text, as sort
works with strings
       // A could possibly be a hash set
      // caseSensitive and removes punctuation
      String line = (caseSensitive) ?
       value.toString() : value.toString().toLowerCase();
      for (String pattern : patternsToSkip) {
       line = line.replaceAll(pattern, "");
      }
      StringTokenizer itr = new StringTokenizer(line,"\n"); // iterate through each
line
      while (itr.hasMoreTokens()) {
       StringTokenizer itr2 = new StringTokenizer(itr.nextToken(), " ");
       // this loop will find a list of unique words in a line
       while(itr2.hasMoreTokens()) {
        String temp = itr2.nextToken();
          if(A.contains(temp) == false && !stopWordsToSkip.contains(temp))
           // check if it stop word to not add
           A.add(temp);
       }
      }
      Collections.sort(A); // in ascending order
      for(int i = 0; i < A.size(); ++i) {
       String wi = A.get(i);
       word.set(wi);
         context.write(word, one); //w[i] emit
       for(int j = i+1; j < A.size(); ++j) {
         word.set(wi+":"+A.get(j));
         // TODO account for converse like hello:world and world:hello
           context.write(word, one);
         //emit pair of w[i] and w[j]
       }
      }
    }
```

```
  } // end of mapper class

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

    private Text word1 = new Text();
    private Text word2 = new Text();
    // for caseSensitive
    private boolean caseSensitive;
    private Configuration conf;
    private BufferedReader fis;

    // key is object as writes to that
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString(), "\n");
      while(itr.hasMoreTokens()) {
        String line = itr.nextToken();
        if(line.contains(":")) { // is a pair
         StringTokenizer itr2 = new StringTokenizer(line, ":");
         String first_pair = itr2.nextToken();
         word1.set(first_pair); // like hello in hello:world 2
         String rest_pair = itr2.nextToken();
         StringTokenizer itr3 = new StringTokenizer(rest_pair, "\t");
         String second_pair = itr3.nextToken();
         String val_int = itr3.nextToken();
         // value has number
         word2.set(second_pair+"|"+val_int);
         context.write(word1,word2);
         // hello:world but also world:hello
         word1.set(second_pair);
         word2.set(first_pair+"|"+val_int);
         context.write(word1,word2);
        }
        else { // not a pair
         StringTokenizer itr1 = new StringTokenizer(line, "\t");
         word1.set(itr1.nextToken());
         word2.set(itr1.nextToken()); // get number 2 like in hello  2
          context.write(word1,word2);
         // write that to context and cast to string
        }
      }
    } // end of mapper function
  } // end of mapper class

  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 class ConfReducer
       extends Reducer<Text,Text,Text,Text> {
    private Text word1 = new Text();
    private Text word2 = new Text();

    public void reduce(Text key, Iterable<Text> values,
                       Context context
                       ) throws IOException, InterruptedException {
      double numerator = 0;
      double denominator = 0;
      boolean set_score = false;
      String second_pair = "";
      String valString;
      ArrayList<String> A = new ArrayList<String>(); // String not Text, as sort
works with strings
      for (Text val : values) {
        A.add(val.toString());
      } // sort this cause values come in shuffled
      // want the single word count to be first
      Collections.sort(A);
      for (String val : A) {
       if(val.contains("|")) {
          StringTokenizer itr = new StringTokenizer(val,"|");
        second_pair = itr.nextToken();
        numerator = Double.parseDouble(itr.nextToken());
        set_score = true;
       }
       else {
          denominator = Double.parseDouble(val);
        }
       if(set_score) { // if both have vals for num and den
          word1.set(key.toString()+":"+second_pair);
        word2.set(String.valueOf(numerator/denominator));
        // set to conf score
          context.write(word1, word2);
        set_score = false;
       }
      }
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    GenericOptionsParser optionParser = new GenericOptionsParser(conf,
       args);
    String[] remainingArgs = optionParser.getRemainingArgs();
    /*if ((remainingArgs.length != 2) && (remainingArgs.length != 5)) {
       System.err.println("Usage: confidence <in> <out> [-skip skipPatternFile]");
       System.exit(2);
    }*/ // TODO put this back later
    Job job = Job.getInstance(conf, "confidence");
    job.setJarByClass(Confidence.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    List<String> otherArgs = new ArrayList<String>();
    for (int i=0; i < remainingArgs.length; ++i) {
       if ("-skip".equals(remainingArgs[i])) {
              job.addCacheFile(new Path(remainingArgs[++i]).toUri());
              job.getConfiguration().setBoolean("confidence.skip.patterns", true);
```

```
                job.getConfiguration().setBoolean("confidence.case.sensitive", false);
            } else {
                otherArgs.add(remainingArgs[i]);
            }
        }
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        // running the second job
        job.waitForCompletion(true) ; // first MapReduce job finishes here
        Configuration confTwo = new Configuration();
        Job job2 = Job.getInstance(confTwo, "Conf step two");
        job2.setJarByClass(Confidence.class);
        job2.setMapperClass(ConfMapper.class);
        job2.setReducerClass(ConfReducer.class);
        job2.setOutputKeyClass(Text.class);
        job2.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job2, new Path(args[2]));
        FileOutputFormat.setOutputPath(job2, new Path(args[3]));
        System.exit(job2.waitForCompletion(true) ? 0 : 1);
    }
}
```