Lab Lecture #3.1

Word frequency in document

This phase is designed in a job whose task is to count the number of words in each of the documents in the input directory. The mapper will receive as input a key that is, by default, the byte offset of the current line in the current file processed (`LongWritable` object) and a value that is the line read from the file (`Text` object). It must output another (key, value) pair. The problem is coding the (word, doc name) pair in a single object. While it is possible to implement a custom class to do the job, we will use a "string trick", emitting a simple string composed by the word, the special character "`@`" and the doc name. In order to obtain the document name from the `Context` object, use the following statement:

```
String fileName = ((FileSplit)context.getInputSplit()).getPath().getName();
```

Then, use the following statements to remove punctuation and other word anomalies:

```
Pattern p = Pattern.compile("\\w+");
Matcher m = p.matcher(value.toString());

while (m.find()) {
      String word = m.group().toLowerCase();
      // remaining code
}
```

During the mapper execution, each word in the line should be lower-cased, and ignored if it does not start with a letter or if it contains the character "_".

The reducer behaves as the standard, well-known `WordCount` reducer. In this case, keys are represented by `Text` objects, and values by `IntWritable` objects. The output will be a set of files (one per reducer): each line of each file will contain a word@document string, a tab character and an integer coded as string. Please remember that Hadoop requires the same classes for keys in the mapper output and the reducer input, as well as the same classes for relative values, although key and value classes can be different.

```java
import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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.input.FileSplit;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordFrequencyInDocument
{
   public static class NewMapper extends Mapper<LongWritable, Text, Text, IntWritable>
   {
      private final static IntWritable one = new IntWritable(1);

      public void map(LongWritable key, Text value, Context context)
         throws IOException, InterruptedException
      {
         // Compile all the words using regex
         Pattern p = Pattern.compile("\\w+");
         Matcher m = p.matcher(value.toString());

         // Get the name of the file from the inputsplit in the context
         String fileName = ((FileSplit) context.getInputSplit()).getPath().getName();

         // Build the values and write pairs through the context
         StringBuilder valueBuilder = new StringBuilder();
         while (m.find()) {
            String matchedKey = m.group().toLowerCase();

            // remove words starting with non letters, digits or containing other chars
            if (!Character.isLetter(matchedKey.charAt(0)) ||
                Character.isDigit(matchedKey.charAt(0)) ||
                matchedKey.contains("_"))
               continue;

            valueBuilder.append(matchedKey);
            valueBuilder.append("@");
            valueBuilder.append(fileName);

            context.write(new Text(valueBuilder.toString()), one);
            valueBuilder.setLength(0);
         }
      }
   }

   public static class NewReducer 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 void main(String[] args) throws Exception
   {
      Configuration conf = new Configuration();
      Job job = new Job(conf, "word frequency in document");
      job.setJarByClass(WordFrequencyInDocument.class);

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

      job.setMapperClass(NewMapper.class);
      job.setCombinerClass(NewReducer.class);
      job.setReducerClass(NewReducer.class);

      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

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