Review of Assignment-2 of @shreysa-sharma

Bishwajeet Dey

October 3, 2017

Report Review

- It would be nice to use Apache Hadoop instead of CDH5.
- The first figure plots the graph with the run number on the x axis and the number of seconds it took to complete the dataset on the y axis. It would be better to plot it against a metric like the serial run or anything else. Since, the run took around 99 minutes, the y axis units could be minutes instead of seconds.
- The second plot probably plots the data from the old dataset and compares serial vs parallel runs. There is a nice anecdote about Amdahl's law. However, It would hold true only if the large dataset was run against the serial version.

Code Review

```
public static class LetterScoreReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

...
 letterOccurances = new TreeMap<>();


 public void cleanup(Context context) throws IOException, InterruptedException {
            long totalCharacters = 0;
            for (Integer val : letterOccurances.values()) {
                totalCharacters += val;
            }

            Integer[] letterScore = new Integer[NUM_CHARACTERS];
            for (int i = 0; i < NUM_CHARACTERS; i++) {
                Character letter = (char) (i + ASCII_START_A);
                float letterOccurance = (float) letterOccurances.get(letter.toString());
                Float percentageOccurrence = letterOccurance / (float) totalCharacters * 100.0F;
                ...

int scoreForLetter = 0;
```

- `CharCountReducer` tries to calculate the final score of all letters in the `cleanup` method. This works only because there is 1 reducer. Also, keeping a `TreeMap` in memory for all character occurrences and outputting final scores could be done differently.

```
public static class LetterScoreMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

  public void map(LongWritable key, Text value, Context context)
```

- `CharCountMapper` can be made more efficient by taking multiple lines at once to emit character frequencies instead of a per line map.

- `LineRecordReader` is a class which tries to to generate a line for the `KNeighborhoodMapper` along with buffer tokens. The class does a lot of work and I am not sure about the correctness of the tokens generated.

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

            int ctr = 0;
            float kMeans = 0.0F;
            for (FloatWritable val: values) {
                if (ctr == 0) {
                    kMeans = val.get();
                } else {
                    kMeans = (kMeans + val.get()) / 2.0F;
                }
            }
            logger.info("key: " + key + " mean: " + kMeans);

            result.set(kMeans);
            context.write(key, result);
}
```

- The reducer does not calculate mean.
- It would be nice to split the classes in their own separate files . A single file of ~630 lines is hard to read/maintain.

```
public static final int NUM_CHARACTERS = 26;
```

- It disregards anything outside the English alphabet.