TI2736-B: Assignment 3 Big Data Processing

Wing Nguyen, 4287118

December 2, 2015

Repository

`https://github.com/codesalad/hahadoop`.
The source code is in the **java/src/** folder.
The output files are in the **java/out/** folder.

1. **Repo:** `java/src/WordCount.java`
   For all of these, the general way to count things is to create an Enum and to increment this during either the mapping or the reduce fase. For example:

```
enum Records { COUNTER; };

public void reduce(Text word, Iterable<Text> filenames, Context context)
                    throws IOException, InterruptedException {
   ...
   context.getCounter(Records.COUNTER).increment(1);
   ...
}
```

   - The number of distinct items are:

```
DISTINCT_TERMS=4674
```

   - Number of words that start with T/t:

```
COUNT_TS=13031
```

   - The number of terms appearing less than 5 times:

```
TERMS_LT5=7206
```

   - The number of files read overall:

```
FILESTOTAL=3
```

   - The 5 most often occurring terms and their frequency in the corpus (can be found in the repo: java/out/):

```
1. the 2906
2. and 2643
3. i   2179
4. to  2162
5. of  1806
```

2. **Repo:** `java/src/OneFileCounter.java`

```
UNIQUE_WORD_FILE=7233
```

3. **Repo:** `java/src/StopWordCount.java`

```
the project 56
the king 53
the world 51
the moore 50
the matter 37
of the 196
of this 96
of my 75
of his 64
of a  55
and the 67
and i 59
and you 36
and then 33
and my 32
```

4. Pseudo-code for queries

   - the minimum temperature ever recorded for each location

```
map(String line):
   String location = line[0];
   int averageTemp = line[3];
   EmitIntermediate(location, averageTemp);

reduce(String locKey, Iterator avgtempValues):
   int lowestTemp = Infinity;
   foreach avgtemp in avgtempValues:
      if (avgtemp < lowestTemp):
         lowestTemp = avgtemp;
   Emit(locKey, lowestTemp);
```

   - the minimum temperature overall in the data set

```
enum Records:
   MIN_TEMP_OVERALL;

mapSetup():
   int minTemp = Infinity;

map(String line):
   String location = line[0];
   int averageTemp = line[3];
   if (averageTemp < minTemp):
      minTemp = averageTemp;
   EmitIntermediate(location, averageTemp);

mapCleanup():
   MIN_TEMP_OVERALL = minTemp;
```

   - for each location, the average temperature per month for all years recorded

```
map(String line):
   String location = line[0];
   int averageTemp = line[3];
   EmitIntermediate(location, averageTemp);

reduce(String locKey, Iterator temps):
   int sum = 0;
   int total = 0;
   foreach int temp in temps:
      sum += temp
      total += 1
   double avg = sum / total;
   Emit(locKey, avg);
```

- for each location, the years for which not a single measurement exists

```
map(String line):
   String location = line[0];
   int year = line[2];
   EmitIntermediate(location, year);

reduceSetup():
   Hashmap<String, List> recorded_years = new Hashmap;

reduce(String locKey, Iterator years):
   foreach year in years:
      recorded_years.put(locKey, year);

reduceCleanup():
   foreach entry<location, years> in recorded_years:
      for (i = 1950 to 2013):
         if (!entry.containsValue(i)):
            Emit(entry.key(), i)
```

5. More pseudo-code for queries

- the list of all questions (specifically their row IDs) that have not been answered by anybody.

```
mapSetup():
   HashSet<Integer> rowIdSet = new HashSet;

map(String line):
   int rowId = line[0]
   int answerId = line[2];

   if (answerId == -1):
      rowIdSet.put(rowId);
   else:
      rowIdSet.remove(answerId);

mapCleanup():
   foreach rowid in rowIdSet:
      EmitIntermediate(rowid, null);

reduce(int key, null):
   Emit(key, nullWritable);
```

- the overall number of users as well as the number of users that have posted questions AND answers

```
enum Records:
   TOTAL_USERS;
   USERS_POSTED_ANSWERED;

map(String line):
   String type = line[1];
   String userId = line[5];
   TOTAL_USERS.increment(1);
   EmitIntermediate(userId, QA);

reduce(String userId, Iterator types):
   HashSet<String> = usertypes = new HashSet;
   foreach type in types:
      usertypes.put(type)
   if (usertypes.size == 2):
      USERS_POSTED_ANSWERED.increment(1);
```

- the id of the question which has received the most answers

```
map(String line):
   int answerId = line[2];
   if (answerId != -1):
      Emit(answerId, 1);

reduceSetup():
   int questionId = 0;
   int totalAnswers = 0;

reduce(int idKey, Iterator counts):
   int sum = 0;
   foreach int count in counts:
      sum += count;
   if (sum > totalAnswers):
      totalAnswers = sum;
      questionId = idKey;

reduceCleanup():
   Emit(questionId, totalAnswers);
```

- the user id of the most active user (most posts, answers or questions)

```
map(String line):
   String userId = line[5];
   EmitIntermediate(userId, 1);

reduceSetup():
   String mostActiveUser = null;
   int mostActiveSum = 0;

reduce(String userId, Interator counts):
   int sum = 0;
   foreach int count in counts:
      sum += count;
   if (sum > mostActiveSum):
      mostActiveSum = sum;
      mostActiveUser = userId;

reduceCleanup():
   Emit(mostActiveUser, mostActiveSum);
```

- the calendar date on which the most questions have been posted.

```
map(String line):
   String type = line[1];
   Date date = line[4];
   if (type == 'Q'):
      EmitIntermediate(date, 1);

reduceSetup():
   Date mostQuestionsDate = null;
   int mostQuestionsSum = 0;

reduce(Date date, Interator counts):
   int sum = 0;
   foreach int count in counts:
      sum += count;
   if (sum > mostQuestionsSum):
      mostQuestionsSum = sum;
      mostQuestionsDate = date;

reduceCleanup():
   Emit(mostQuestionsDate, mostQuestionsSum);
```