CSCI4180 Tutorial-4

Assignment 1 (Hints)

ZHANG, Mi

mzhang@cse.cuhk.edu.hk

Oct. 15, 2015

Assignment 1

- Due on Oct. 22
- Configure VMs & Azure platform
  - Install Hadoop successfully.
  - Run WordCount program on your Hadoop platform.
- Write Java Program
  - Word Length Count
  - N-gram Count
  - N-gram Relative Frequency
- Test on the KJV & Shakespeare data
- Do some optimizations

Part 2: Word Length Count (20%)

- Optimize the WordCount program with in-mapper combining
  - Combine the output of mapper before emitting to reducer.
  - ✓ Save traffic.
- You may use an associative array to count the occurrences of the length of words
  - In Java, you may use a HashMap: http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html
  - What should be the type parameters of the HashMap?
    - Hint: What are the data types of length and count?
    - Output: Each line contains a tuple of (length, count).
    - Suggestion: The output is separated by a space character.

Part 2: Word Length Count (20%)

```
public static class Map extends Mapper<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
   // Define variables or methods here if necessary
   protected void setup(Context context) {
      // This method will be executed exactly ONCE
      // at the beginning of the Map task
   }
   protected void cleanup(Context context) {
      // This method will be executed exactly ONCE
      // at the end of the Map task
   }
   protected void map(KEY_IN key, VAL_IN val, Context context) {
      // Take the input (key, val) for the job
      // Execute many times
   }
}
```

Initialize the HashMap H here

For all item *t* in H
`Emit(t,H(t))`

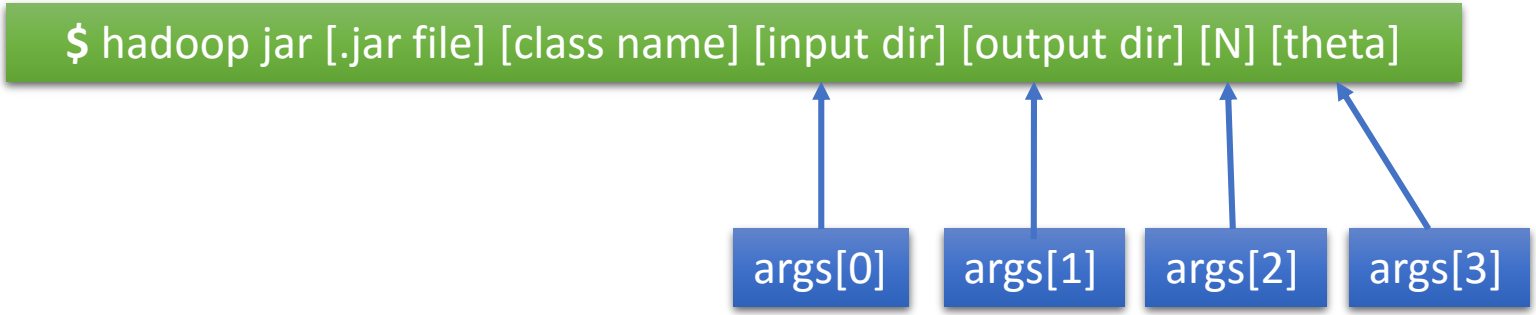
Add each key-value pairs into H.
Pseudocode:
H(val) ← H(val) + 1

Part 3: N-gram Initials Count (25%)

- Reminder:
  - ONLY output the count of the initial sequences that are all in alphabet.
  - The initials are case-sensitive (e.g., “A b” and “a b” are different).
  - The word in the end of a line and the word in the beginning of the next line ALSO form an N-gram.
- Idea:
  - Add a class variable (i.e. declared static) to record the last N-1 word’s initial characters.
  - The key to emit in the mapper can be a string concatenating the N initials.
  - E.g., N=3, emitting (key: “abc”, value: count of “a b c”)

Part 4: Count N-gram Initials RF(25%)

- Reminder:
  - Relative frequency = COUNT(“X Y Z”) / COUNT(“X *”)
    - Here N = 3, * stands for any ALPHABET initials.
  - Output:
    - Each line contains a tuple of (1st word’s initial, 2nd word’s initial, … , N-th word’s initial, frequency), separated by a space character.
    - ONLY output tuples with frequency ≥ θ
- Parameter passing format:



  - How to input the command line arguments?

Part 4: Count N-gram Initials RF(25%)

- To pass θ to the MapReduce framework:

```
Configuration conf = new Configuration();
Job job = new Job( conf, “ngraminitialrf" );
conf.set( "threshold", args[3] );
```

Inside `main()`

- To retrieve and parse the value of θ inside the reducer:

```
double threshold = Double.parseDouble(
        context.getConfiguration().get( "threshold" )
);
```

Inside `reduce()` of the reducer

Submission Guidelines

- You must at least submit the following files, though you may submit additional files that are needed:
  - WordLengthCount.java
  - NgramInitialCount.java
  - NgramInitialRF.java
  - Please strictly follow the file names!
- The testing platform during demo is provided by the TAs
  - If you need to change the configuration of Hadoop, make sure that you do it dynamically inside the code (see P.38 of lec3.pdf).
  - You are not allowed to modify any configuration files during demo.

Remarks

- We provided two sample dataset for your development:
  - KJV Bible
  - The complete works of Shakespeare
- You may encounter a problem when using the Shakespeare data because of the directory structures
  - MapReduce does not support inputting data in folder recursively.
  - To solve this problem, you can move all the .txt files to the same directory and use this directory as the input.
  - Alternatively, you can append all the contents in all the .txt files to a single file.
    - This approach also results in shorter execution time because of the number of mappers is smaller.

Remarks

- Do your assignment as early as possible! You (and the TAs) never know what will happen in the future.
  - The TAs cannot guarantee that they will be available to answer your questions and handle bugs on VMs in the last few hours…
- Please test your program with large dataset in order to discover potential problems in memory and performance.
- To verify your program's correctness, you can use something you are familiar with to generate sample output.

Bonus(5%, optional)

- The top 3 groups whose Part 4's programs have the smallest running time can receive the bonus marks
- Prerequisite: Your Part 4 program returns correct answers.
- What can you do?
  - Optimize the program by modifying the algorithm or designing more efficient data structures.
  - Configure some parameters in Hadoop.
  - Remember you need to do this in your program dynamically, not by modifying the configuration files.

Bonus(5%, optional)

- Hints:
  - MapWritable is bad! In Java, the simpler the data structure, the more efficient the program is. So, what should you use?
  - What are the possible values (ranges) of the keys?
  - Are there any optimization techniques that can be applied to your program?
  - How to minimize the traffic?
  - Is it possible to reuse some data structures in your program?
- I don’t have “model” answers to these questions.
  - Try to be creative
  - Try to ask yourself more questions! They may help you optimize your program.

Thank you!