CS435: Introduction to Big Data

GTA: Bibek R. Shrestha
Email: cs435@cs.colostate.edu

Feb 16, 2018
Today...

- Discussion on Programming Assignment 1
Dataset

- ~1.5 million Wikipedia articles (summarized)
- ~1 GB size
- Format:
  
  `Title_of_Article-1<=====>DocumentID<=====>Text_of_Article-1`
  
  `Title_of_Article-2<=====>DocumentID<=====>Text_of_Article-2`

Notes:

- *Text of an article starts with its title.*
- *Unigrams are generated on Text_of_Articles only.*
Profile 1

- A complete list of unigrams in the dataset
- Must be sorted in alphabetical order (ascending)
- Eliminate duplicates.
- Use of combiner to eliminate local duplicates can be important (one of the approaches that can be taken).
Profile 1 Solution

- This is very similar to WordCount program except you do not have to count the word frequency in the reducer.
- Emit the key received from the mapper function.
- Mapper function emits (word, 1) or (word, null).
- In Reducer, `context.write(key, null)`
- Why do we need Reducer then? ->For sorting the mapper output.
- If number of reducer more than 1, as defined using `job.setNumReduceTasks(5)` in driver program, we will get sorted output per reducer, not the globally sorted output as required!
- Use Partitioner to get globally sorted output (defined in driver):
  `job.setPartitionerClass(profile1Partitioner.class)`
Profile 1 Solution

• **Partitioner code snippet:**

```java
@Override
public int getPartition(Text key, IntWritable value, int numReduceTasks){
    if (numReduceTasks == 5){
        Character partitionKey = key.toString().toLowerCase().charAt(0);
        if (partitionKey >= 'a' && partitionKey <= 'd')
            return 0;
        else if (partitionKey >= 'e' && partitionKey <='k')
            return 1;
        …..
    }
}
```
Profile 2

- A list of unigrams and their frequencies within each article
- Similar to word-count program
- Each article has unique integral Document ID.
- Output should be grouped by Document ID.
- Output files will correspond to the number of reducer used.
Profile 2 Solution

- Use composite key (natural key and natural value) i.e. containing both (DocumentID and unigram).
- Use SortComparator to order by composite key.
- Use Partitioner and grouping comparator on natural key for partitioning and grouping.
Profile 3

- List of unigrams and their frequencies in the corpus
- List should be sorted from most frequent unigrams to least frequent ones.
- Solution to generate profile 3 can be seen as combination of profile 1 and 2.
- Here, we list unigram with its total occurrence in the complete dataset (of course, without considering per article construct as of profile 2).
Profile 3 Solution

- Similar to Profile 1
- Excepts count the frequency in reducer as you did in WordCount program in PA0
- And, sorting is based on the value of (key, value) pair output from reducer
- So you need second mapreduce job.
- And, to sort by value, reverse key, value pair being sent by first reducer job.
Preprocessing

• Code Snippet in mapper (Profile1):

    if (!value.toString().isEmpty()) {
        StringTokenizer token = new StringTokenizer(value.toString().split("<=====>")[2]);
        while (token.hasMoreTokens()) {
            String out = token.nextToken().replaceAll("[^A-Za-z0-9]","").toLowerCase();
        }
    
    ....
In the next recitation...

- We will start Programming Assignment 2