Lab 2 Submission Instructions

CSE 168 • Fall, 2022

Choices

For Lab 2 you can choose any one of the following tasks to complete using MapReduce.

Option 1: Anagram

Write a Map/Reduce program in Hadoop to process a file with a list of words and identify all the anagrams in the file. Examples:

- drawbacks, backwards
- cheatable, teachable
- hills, shill
- rebroadcast, broadcaster

The input files you can test your code on are located under 'Files' → 'Lab 2' → 'anagram-input-files'. There are three files under this folder, namely 'file0', 'file1', and 'SINGLE.txt' which will allow you to test your code. 'file0' contains the example discussed in slides and contain only one anagram (hills, shill) whereas 'file1' contains few more anagrams. Based on the instructions to run Hadoop (shared as another doc), you can test your program first with 'file0' first and then with 'file1'. And after that you can test it with 'SINGLE.txt' file which is a larger file. Feel free to create more test cases by your own and test out your program. (Note: 'outputCollector' used in the slides was running older version of Hadoop and hence you will not be able to use it with the current version we are using for lab. You can instead use 'context' as used in WordCount example program)

Option 2: Data clean (Homework 3)

You are given a file with keys and values. The keys are in increasing order from 1 to N, but some keys in this file may be missing. For convenience, we assume that key 1 will always exist. The goal is to reconstruct missing keys with the value from the previous key.

For example, if the file contains: `(1, A), (2, B), (4, C)`

The cleaned output would be: `(1, A), (2, B), (3, B), (4, C)`

Assume that only one key may be missing in each gap. That is, if key k is missing, key $k-1$ always exists. Produce a MapReduce program that will clean the above dataset.

The input files you can test your code on are located under 'Files' → 'Lab 2' → 'dataclean-input-files'. There are three files under this folder which will allow you to test your code- 'file0', 'file1', or 'file2'. Based on the instructions to run Hadoop (shared as another doc), you can test your program first with 'file0', then with 'file1', and lastly with 'file2'. Feel free to create more test cases by your own and test out your program.

Option 3: Term Frequency

Term Frequency (TF) is the number of times a particular word appears in a document. TF-Inverse Document Frequency (IDF) is widely used in text analysis and information retrieval tasks. For this lab task, we are only focusing on computing the TF and not IDF. Write a map/reduce program in Hadoop to find out the term frequency for all the words in the document. The input file, 'file0' you can test your code on is located under 'Files' → 'Lab 2' → 'termfreq-input-files'. The contents of 'file0' are as follows:

```
1,i love dogs

2,i hate dogs and knitting

3,knitting is my hobby and my passion
```

'file0' contains three documents with ids 1,2, and 3 and each document is made up of a sentence. The output of the your program should look contain 'token' 'document id', 'term frequency'

For example, for the input above the output would be:

```
dogs     1, 1
i        1, 1
love     1, 1
and      2, 1
dogs     2, 1
hate     2, 1
i        2, 1
knitting         2, 1
and      3, 1
hobby    3, 1
is       3, 1
knitting         3, 1
my       3, 2
passion  3, 1
```

You can read through this article to further understand Term Frequency

Submission instructions for whichever program you choose

You need to submit an archive file (only .tar.gz/.zip extensions allowed) containing the following:

1. JAR file of your program (i.e., either Anagram.jar or DataClean.jar or TermFreq.jar)
2. Your source code as individual files containing your MapReduce code (from either Anagram.java or DataClean.java or TermFreq.java)
3. A PDF document containing a brief explanation about how you went about writing the map and reduce functions for the program of your choice.

On the lab machines you can generate the archive file using the following command:

```
tar -cvf firstname-lastname-dataclean.tar.gz DataClean.jar DataClean.pdf
```

Hence, if you worked on data cleaning, you need to submit **firstname-lastname-dataclean.tar.gz** or **firstname-lastname-dataclean.zip**. Make sure there are no errors when you run your JAR file on Hadoop and your program works as intended. If there are errors while we run your JAR or your program does not run as intended, points would be deducted.