Hadoop Traffic Analysis

CSInParallel Project

August 11, 2014

CONTENTS

1 Contents: 2
1.1 Introduction to the Dataset 2
1.2 Working with Multiple files 4

This module was created for CSInParallel by Jeffrey Lyman in 2014 (JLyman@macalester.edu)

The purpose of this module is to teach students how to analyze datasets distributed over multiple files using the Hadoop framework. It is assumed that students are already familiar with the basics of hadoop and CSInParallel's Web Map Reduce hadoop interface.

The excersises in this module use a dataset from the UK department of Transportaion that contains detailed records of traffic accidents split into three separate files.

The dataset can be obtained from Academic Torrents. More about the source of this public data from the United Kingdom can be found on its Wikipedia page.

CHAPTER

ONE

CONTENTS:

1.1 Introduction to the Dataset

1.1.1 The Data

The UK department of Transportation keeps detailed records of all traffic incidents. Fortunately for us they made this data available to the public in the form of three csv files that contain information about the accidents, casualties, and vehicles involved.

System-dependent Alert

The path of the dataset shown below may not be the same on your WMR system. It is correct for this WMR server:
selkie.macalester.edu/wmr

These files are located on wmr in the `/shared/traffic` folder and are named `Accidents7409.csv`, `Casualty7409.csv and Vehicles7409.csv` respectively.

1.1.2 Working with the Data

Each line in the files contains several fields seperated by commas, to accesss these values, it is necessary to call `key.split(’,’)` (or the equivalent in whatever language you’re using) to get an array of values. If you want, you can turn these values into an object, however it’s faster to simply refer to them by their index

index	Accidents7409.csv	Casualty7409.csv	Vehicles7409.csv
0	Accident Index	Accident Index	Accident Index
1	Location Easting OSGR	Vehicle Reference	Vehicle Reference
2	Location Northing OSGR	Casualty Reference	Vehicle Type
3	Longitude	Casualty Class	Towing/Articulation
4	Latitude	Sex of Casualty	Vehicle Maneuver
5	Police Force	Age Band of Casualty	Vehicle Location Restricted Lane
6	Accident Severity	Casualty Severity	Junction Location
7	Number of Vehicles	Pedestrian Location	Skidding/Overturning
8	Number of Casualties	Pedestrian Movement	Hit Object in Driveway
9	Date	Car Passenger	Vehicle Leaving Driveway
10	Day of Week	Bus/Coach Passenger	Hit Object off Driveway
11	Time	Pedestrian Road Maintenance Worker	1st Point of Impact

Continued on next page

Table 1.1 – continued from previous page

index	Accidents7409.csv	Casualty7409.csv	Vehicles7409.csv
12	Local Authority (District)	Casualty Type	Was Vehicle Left Hand Drive
13	Local Authority (Highway)	Casualty Home Area Type	Journey Purpose of Driver
14	1st Road Class		Sex of Driver
15	1st Road Number		Age Band of Driver
16	Road Type		Engine Capacity
17	Speed Limit		Propulsion Code
18	Junction Detail		Age of Vehicle
19	Junction Control		Driver IMD Decile
20	2nd Road Class		Driver Home Area Type
21	2nd Road Number		
22	Pedestrian Crossing Human Control		
23	Pedestrian Crossing Physical Facilities		
24	Light Conditions		
25	Weather Conditions		
26	Road Surface Conditions		
27	Special Conditions		
28	Carriage Hazards		
29	Urban or Rural Area		
30	Did Police Officer Attend Scene		
31	LSOA of Accident Location		

Most of the values are determined by special codes which which can be found in the pages of `this spreadsheet`

1.1.3 Example Job

Let’s use what we’ve learned to answer a quick question. Between 1974 and 2004 were there more casualties per incident in rural or urban accidents?

Our `mapper` will need to emit a key that represents whether the accident was rural or urban and the number of casualties as the value.

Our `reducer` will need to sum the casualties for each type of accident and divide them by the total number of accidents.

Given that the code that tells whether a crash was urban or rural is stored at index 29 of the accident csv and the number of casualties is stored at index 8 our code looks like this:

```
def mapper(key, value):
  data = key.split(’,’)
  casualties = data[8]
  urbanOrRural = data[29]
  Wmr.emit(urbanOrRural, casualties)

def reducer(key, values):
  count = 0
  total = 0
  for value in values:
    total += int(value)
    count += 1
  Wmr.emit(key, total / count)
```

Note: Does this reducer look familiar?

Run this job on wmr using cluster path /shared/traffic/Accidents7904.csv You should get the following output:

1	1.2805146224316546
2	1.5105844913989401
3	1.4071045576407506
-1	1.3062582787269292

A quick glance at the spreadsheet reveals that 1 stands for Urban, 2 for rural, and 3 for unallocated. -1 means that neither was reported. It appears that on average rural accidents tend to involve more casualties.

1.2 Working with Multiple files

The sample question from the last section was fairly simple to answer because all of the data could be found in one file. However data is often split between files, making it harder to process.

Take this question for instance: are taxis more likely to get into crashes on the weekend?

1.2.1 Taxi Crashes

To answer this question we will need to access the day of week data at accidents[10] and the vehicle type data at vehicles[2] (codes 8 and 108 represent taxis). However those two bits of data are in two seperate files so we'll need some way to cross reference them. We'll do that with the accident index stored at accidents[0] and vehicles[0]

This also means that we'll need to access multiple files during a single job. Luckily WMR makes this easy for us. If we enter a folder into the cluster path, it will use all the files in that folder has input.

However we still need to be able to tell if a mapper key came from the accidents file or the vehicles file. We can do this by looking at the length of the data list. The Vehicles file has 21 peices of information while the Accidents file has 32. Armed with this information we can write a mapper and a reducer that will filter out accidents based on whether they involved a taxi. Run this code using Cluster Path /shared/traffic

```
def mapper(key, value):
    data = key.split(’,’)
    if len(data) == 21:                   #vehicle data
        if data[2] in (’8’, ’108’):       #codes for taxis
            Wmr.emit(data[0], "taxi")
    elif len(data) == 32:                 #accident data
        Wmr.emit(data[0], data[10])
```

This mapper checks to see whether input came from accident data or vehicle data. Then, if it was accident data, it emits the day of the week that the accident occured on. If it came from the vehicles data then it emits a message if a vehicle involved was a taxi.

Our reducer takes that output and emits a list of accident indices and the day of the week that they occured on.

```
def reducer(key, values):
    isTaxi = False
    dayOfWeek = ""
    for value in values:
        if value == "taxi":
            isTaxi = True
        else:
            dayOfWeek = value
    Wmr.emit(dayOfWeek, key)
```

This works because only one day of week value is emitted per accident index and while there can be more than one taxi involved in a given crash.

But we're not done yet. We simply have list of crashes and a list of the days on which they occured. We still need to count them.

We can this by using the output of the last job to run a new job. Just hit the use output button at the top or bottom of the page.

Our mapper will recieve days of the week as keys and ones as the values. We just need to feed these straight into a `counting reducer` by using what's known as the `identity mapper` our code is as follows:

```
def mapper(key, value):
    Wmr.emit(key, value)
```

```
def reducer(key, values)
    count = 0
    for value in values:
        count += int(value)
    emit(key, count)
```

After submitting the job on WMR we get the following output:

1	693847
2	873422
3	877086
4	890605
5	934161
6	1058859
7	896218

Code 1 is Sunday, code 2 is Monday etc. So it looks like Taxis get into the most accidents on Fridays, a fairly high number on Saturdays, but very few on Sundays.

1.2.2 Challenges

Use the techniques you've learned to answer the following questions, or come up with your own:

- Are male drivers more likely to injure other males? You will need the following fields: Sex of the driver - Vehicles[14], Sex of casualty - Casualties[4] in both cases 1 is male 2 is female 3 is unknown and -1 is missing data.
- What is the average severity of a crash in which at least one vehicle overturned? If vehicles[7] = 2, 5, or 4 the vehicle overturned. The severity of an accident is Accidents[6] and ranges from 1-3, 1 being the most serious.
- Are trucks more deadly than vans?
- Create a graph showing the number of traffic accidents at each hour of the day. If you're feeling adventurous seperate it out by day and hour.
- Devise some of your own questions to ask of this data.