

## Overview:

For this practical, we had to utilise Hadoop to read Twitter Json files, and evaluate the frequency of various hashtags. This at first seemed relatively challenging, however simply required an understanding of the problem at hand, and the format of the input files.

## Design & Implementation:

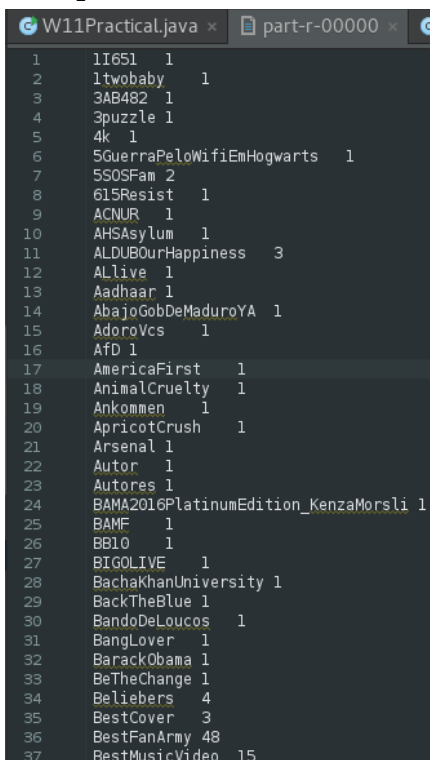
The only code that I actually created for the practical was the findHashTags method, to create this, I copied to supplied example to ensure that it fit within the supplied Hadoop structure, then changed the output so instead of simply reading every word, it read through the Json entity, and output any hashtags found. To find exactly where they were stored, I had to evaluate the Twitter API supplied, then it became a question of reading through the Json Entity, ensuring that each key existed, then getting the value associated with said key. I also implemented a small method to delete the input file before creating the Hadoop output folder, this could potentially be dangerous, so in the basic file I have commented it out, to avoid unwanted file deletions.

The Reduce method was exactly the same, and unchanged.

## Testing:

Testing mainly involved running the method, and then reading through the output file, then deleting it and retrying with alterations until the program passed stacscheck. No edge cases occurred to me, so I checked if the file was valid Json, but other than that. Very little testing of the base file took place. I also created a seperate Json object called Tweet which was one tweet just reformatted, it wouldn't pass into the object, but it worked as a helpful guide to see where objects where. I also ran the program on several of the files from studres, just to ensure that it was not a fluke that it worked.

## Examples:



```
1 1I651 1
2 1twobaby 1
3 3AB482 1
4 3puzzle 1
5 4k 1
6 5GuerraPeloWifiEmHogwarts 1
7 5SOSFam 2
8 61SResist 1
9 ACNUR 1
10 AHSAsylum 1
11 ALDUBOurHappiness 3
12 ALLive 1
13 Aadhaar 1
14 AbajoGobDeMaduroYA 1
15 AdoroVcs 1
16 Afd 1
17 AmericaFirst 1
18 AnimalCruelty 1
19 Ankommen 1
20 ApricotCrush 1
21 Arsenal 1
22 Autor 1
23 Autores 1
24 BAMA2016PlatinumEdition_KenzaMorsli 1
25 BAMF 1
26 BB10 1
27 BIGOLIVE 1
28 BachaKhanUniversity 1
29 BackTheBlue 1
30 BandoDeLoucos 1
31 BangLover 1
32 BarackObama 1
33 BeTheChange 1
34 Beliebers 4
35 BestCover 3
36 BestFanArmy 48
37 BestMusicVideo 15
```

An example of the output from the program, producing the tweets, this appears to be identical to the desired output.

### **Evaluation:**

The program runs smoothly, and the code is simple and readable, any possible exceptions are caught and any bugs are handled, no file seems to cause any unexpected behaviour.

### **Conclusion:**

I believe that the basic deliverable for this practical was fairly easy to obtain, and while I should possibly have done some more testing on my program, I decided that the tests that I had run it through worked, and the program didn't seem complicated enough to encounter obscure bugs, so I focused the majority of my efforts on the extensions.

### **Extensions:**

#### **Overview:**

For the extensions I wanted to give the user a more friendly interface, being able to decide which calculation they wanted to run on their data, initially I created a method to get the user input and supply them with a choice of 3 options, then have them input a selection, however I decided it would be more user friendly to have a graphical interface. The benefit of this was that the user didn't have to add any arguments, and instead could just input their input and output file in the text boxes as desired, then press a button, then change the file directories, and do it all again.

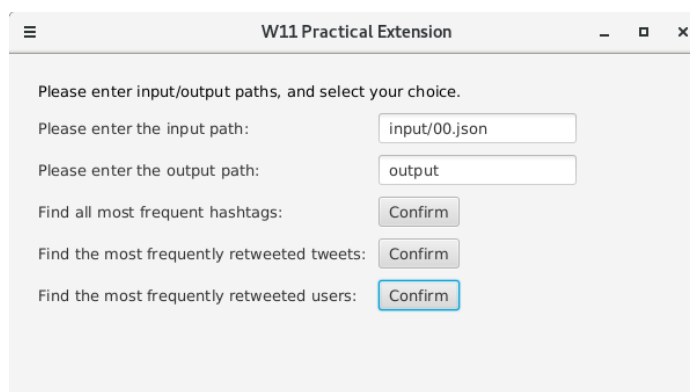
### **Design & Implementation:**

The interface was achieved using JavaFX, where the user could simply input their input and output path, and then press a button to initiate a specific map-reduce job on the specified data. Since two of the jobs involved sorting in terms of most popular, I decided to remove a tiny bit of functionality, but forcing the hashtags to be sorted in order of popularity. I also encountered a bug, whereby Hadoop could not handle the entire Tweet text, so I shortened it down to just the unique id. While this is less user friendly, since it does not show the information of the tweet, it was the only way I could see of doing it. I also added in running the jobs on multiple cores, but this did not appear to affect the programs runtime in anyway, occasionally making it run slower.

### **Testing:**

To test, I ran my program, then looked at the output, and compared it to the input data, checking numbers, and how likely certain things appeared, I also added a lot of print statements to decide where errors were occurring. This was mainly useful as Map and Reduce methods would not fire, and not give any indication of not firing, so the print statement aided.

### **Examples:**



The screenshot shows a JavaFX window titled "W11 Practical Extension". Inside the window, there is a text label "Please enter input/output paths, and select your choice." followed by two text input fields. The first field is labeled "Please enter the input path:" and contains the text "input/00.json". The second field is labeled "Please enter the output path:" and contains the text "output". Below these fields are three buttons, each with a "Confirm" label. The first button is for "Find all most frequent hashtags:", the second for "Find the most frequently retweeted tweets:", and the third for "Find the most frequently retweeted users:". The third button is highlighted with a blue border.

## **CS1003 W11 Practical 160006128 Tutor: Kasim Terzić 21 May 2016**

This is an example of the GUI, achieved using JavaFX. The program will remain open, until manually closed, and features the 3 jobs outlined in the specifications as extensions.

### **Evaluation:**

I feel that a user interface was almost entirely unnecessary for this practical, however I enjoyed implementing it, and it certainly helps with making the program more simple and easy to run. I am mildly disappointed that I was forced to have the tweets only show up as unique ids, however, when the tweets were showing up fully, the program was incredibly messy, given the variation in formatting. The code itself is readable and well documented, with the basic deliverable passing all the stacschecks style checkers.

### **Conclusion:**

I believe that my program does all that was asked of it. Given more time, I would have liked to add more analysis of the twitter data, and possibly beautify the output to make it more readable.