COMP1204 - Coursework 1: Unix

Daniel Best (Student ID: 29777127)

March 7, 2019

1 Scripts

1.1 Basic File Processing (countreviews.sh)

The first task of this coursework was to perform some basic file processing on the TripAdvisor data. Specifically, this involved retrieving the number of reviews for each hotel file, and then outputting a sorted list of hotels, ordered by the most reviews to least.

The script file designed for this task, **countreviews.sh**, can be seen below:

```
#!/bin/bash

for file in $1/*; do
       echo "$(basename $file .dat) $(grep -c "<Author>" $file)"
done | sort -n -r -k 2
```

Listing 1: countreviews.sh

This script loops through each **file** in given file path, which is represented as **$1/*** (all files found in the passed file path argument) in the code. For each file, the script outputs the **basename**, i.e. the name of the file without the file path or extension, and the number of **<Author>** tags in the file, which represents the amount of reviews. This second value is found by using the **grep** command, with the **-c** argument specifying that the number of instances of the search query found should be returned, as opposed to the usual output of the found instances themselves.

Once the for loop has been completed, the output is piped into the **sort** command, which sorts the results in numerical order (**-n**), in reverse order (**-r**), and based on the second column (**-k 2**).

1.2 Data Analysis (averagereviews.sh)

The other scripting task for this coursework was to perform data analysis on the same TripAdvisor dataset. In particular, the task was to get the average overall rating for each hotel by calculating the mean of all the **<Overall>** values found in the file.

The **averagereviews.sh** script file was created to perform this task, and can be seen below:

```
#!/bin/bash

for file in $1/*; do
        total=0
        i=0
        for rating in $(grep "<Overall>" $file | sed 's/^.*<
            Overall>//; s/\r$//'); do
               total=$((total + rating))
               i=$((i + 1));
        done
        name=$(basename $file .dat)
        echo | awk -v name="$name" -v total="$total" -v i="$i" '{
            printf "%s %.2f\n", name, total/i}';
done | sort -n -r -k 2
```

Listing 2: averagereviews.sh

This script also loops through each **file** in the passed file path argument, as represented by **$1/***. However, the script then enters another for loop which finds each **<Overall>** value in the file and adds this to the **total** value, whilst also incrementing the **i** value by one when one such value is found. This is achieved by means of a **grep** command that finds all such tags in the file, with this result being piped into a **sed** command which strips away the **<Overall>** tag and the carriage return character at the end of the line, which would otherwise cause issues for the arithmetic operation that follows.

Once this information is collected, the **basename** value is of the file is found and these three variables are given as arguments to an **awk** command; this command simply outputs the hotel name followed by the calculated mean rating value (*2dp*), which is calculated by dividing the **total** value by the **i** (number of iterations) value. It is worth noting that an **echo** command is needed before the **awk** command to generate an output on the console. It is also worth noting that the **awk** command was also only used as the **bc** command, which is designed for floating point arithmetic, was not installed on the COMP1204 Debian VM.

As with the previous script, this output is then piped into a sort command that sorts the results in numerical order (**-n**), in reverse order (**-r**), and based on the second column (**-k 2**).

2 Discussion

This final section will analyse the problems that TripAdvisor may face by collecting reviews in this manner.

Currently, this information is stored in an unstructured markup file that makes performing detailed analysis much more difficult than it needs to be. The tags in the current file do act as meta-data to an extent, but the files as they are right now are more like glorified **.txt** files as opposed to a structured markup file like **XML**. This kind of storage type is sufficient for small amounts of data, but less suited for the large amount of data that TripAdvisor has for its reviews.

At the very least, this data should be stored in **XML** files that would allow for the use of tools like **XPath** to make performing analysis easier. Ideally however, this data would instead be moved into a structured database, where **SQL** or another querying language could be used to effortlessly perform the exact same analysis.

The way that the reviews are collected and ranked can also be improved. To this end, the following are ideas as to how TripAdvisor could go about doing this:

1. Temp