COMP1204 - Coursework 1: Unix

Daniel Best (Student ID: 29777127)

March 11, 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 the given file path, which is represented as **$1/*** (all files found in the passed file path argument). For each file, the script outputs the **basename**, i.e. the name of the file without the file path or extension, as well as 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 the returned value, 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 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 $file .dat)
        awk -v name="$name" -v total="$total" -v i="$i" 'BEGIN{
            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 sums these in the **total** value, whilst also incrementing the **i** value by one at each iteration. 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 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 the **awk** command was 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 metadata 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.

A **database** will, however, bring the disadvantage of requiring slightly more storage than the current **flat-file** structure for its actual underlying structure, but it would also allow for more advanced manipulation of the data, such as creating **tables** and **views**. It will also have the ability to easily record any **transaction** that may occur between a particular user and the data. A **database** also has the advantage of being easily transferable due to the presence of a **Database Management System (DBMS)** that defines how the data is structured and how it can be interacted with.

In general, the **flat-file** structure that TripAdvisor utilises may very well lead to **data duplication** - for instance: if a user has two reviews, then the data of their name must be duplicated for both records. In a **database**, the user data would exist in its own table, which is then simply referenced by another table that contains the actual review data. This would achieve **data redundancy** and likely result in the **database** requiring less storage space than the current **flat-file** data, despite the need for more initial storage for the database's **schema** information and the **DBMS**.

Apart from the way in which the data is stored, the manner in which the reviews are collected and ranked can also be improved. To this end, TripAdvisor could make use of the following ideas:

1. After having analysed some of the data, it seems that the **Overall** rating is user-defined as opposed to being calculated by some kind of mathematical average by the system. It would make more sense to allow the user to select the score for other rating fields, and then calculate an **Overall** value based on those values.
2. Currently the user is able to leave some of the rating fields blank, with the exception of the **Overall** field. By forcing the user to populate each of these fields, the quality of each review should significantly improve.
3. Reviews could be made obsolete after a particular duration has passed, as a review that was helpful a significant amount of years ago may now be irrelevant, and will almost certainly be less informative than a review made more recently.

As well as the way in which they collect reviews, TripAdvisor should also consider making changes to the current system in order to increase the validity of these said reviews. Here are some proposals as to how they could go about achieving this:

1. An additional check could be added to verify that the **IP Address** of a new reviewer differs from previous reviewers in order to stop one individual from posting lots of fraudulent reviews about a particular hotel.
2. Some reviews contain little to no comments, and therefore it is difficult to tell how a user justifies each score given in a review. TripAdvisor could consider a specific comments field for each individual rating to help justify each score the reviewer gives, which should allow them to manually determine untrustworthy reviews more easily.
3. In order to ensure that less fraudulent reviews are committed, a reviewer could first be made to connect their TripAdvisor account with some form of **social media** account such as Facebook, Snapchat or Whatsapp. This should make it more difficult to set up a fake account just to create a bias review.