Unix Coursework

Ioana Alexandra Nesteriuc

March 2018

School of Electronics and Computer Science

University of Southampton

Student ID: 29218179

Contents

1 Scripts **3**
1.1 Basic file processing: countreviews.sh 3
1.2 Data Analysis: averagereviews.sh 4

2 Discussion **5**

1 Scripts

1.1 Basic file processing: countreviews.sh

```
#!/bin/bash

reviewsfolder=$1

for i in $1/*
do
reviewsnumber=$(grep -Ec "<Author>" $i)
echo $(basename -s .dat $i) $reviewsnumber
done | sort -k2rn
```

The first line represents a shebang which will tell the shell how to interpret the script when it is executed. In our case, the scipt will be run by bash shell.

The next line will store the value of the path sent as argument in a variable called "reviewsfolder". Even though this variable is not necessary, its main purpose is to make the code easier to understand and, therefore, more maintainable.

The script will then iterate over all the files in the given folder and for each file it will determine the number of reviews. To do so, the grep command is given a markup text pattern that occurs for every review and also two additional flags:

1. -E: this will include all of the basic meta-characters and some additional meta-characters which will minimize the possibility to consider a result that is within the actual review
2. -c: this will count the number of instances the given pattern was found within that file and will return this number

Finally, we will print the base name of all the files and the number of reviews, but this does not happen yet as the loop output is piped to the sort command. This will numerically sort the output by the second column and will reverse the order due to the following options:

1. -k2: specifies that the sort is by the second column
2. -n: specifies that the sort will be numerical
3. -r: reverses the default order

1.2 Data Analysis: averagereviews.sh

```
#!/bin/bash

reviewsfolder=$1

for i in $1/*.dat
do
thisfile=$(basename -s .dat $i)
grep "<Overall>" $i | awk -v filename=$thisfile -F">"
'BEGIN{sum=0;count=0}
{sum+=$2;count++}
END{printf("%s %.2f\n", filename, sum/count)}'
done | sort -k2rn
```

The averagereviews.sh script will iterate over all the files of type dat that are in the folder sent as argument. For each file, it will save its base name into a variable called "thisfile", without storing the .dat extension due to the -s option.

Then, for each file, the script searches using the grep command for a markup text pattern that is specific to every review in the file (in our case "<Overall >") and will pipe the output to the awk command. This command will use as a field separator the symbol >and will then determine the sum of all reviews and the number of the reviews. Since the review value is in the second field, we can access this value using the dollar symbol and the value 2.

Finally, the awk command prints the base name of the file as specified above and the division result with two decimals. Because the loop is piped to the sort command, the print will only take place after the sort command is executed. The output will be sorted by the second column (-k2), numerically (-n) and in the reverse order (-r).

2 Discussion

One of the main challenges that TripAdvisor faces in collecting reviews in this way is processing data in a time-efficient way. Having to iterate over every line in all the hotel files in order to find data about a specific topic implies doing many time-consuming operations. If, instead, the company opted to store the data in a database, they could process the number of reviews and determine the ranking using simple queries which would most likely take linear time.

Online data is very dynamic and, therefore, it must require a large amount of space when it is stored. Storing data in files not only takes more space, but also facilitates the occurrence of data redundancy. For instance, if the same user decided to write two identical reviews (and was able to do so), in the file system there would be two identical reviews, whereas in a database one would overwrite the other.

At the same time, even if there is a very small possibility that the pattern given to the grep command matches text from the reviews, there is no guarantee that such event can not occur, which implies that in that case the script will either fail, either output altered results.

Ensuring that the reviews are trustworthy is a challenging task as there exists no tool that can absolutely guarantee that one review is sincere or not. However, the company can ensure that the user was indeed a customer of the hotel they are reviewing by matching data from TripAdvisor's file system to data from the hotel's file system/database. If at least one of the details from the hotel's file system/database correspond to those that TripAdvisor have (full name, date of birth, National Insurance Number, phone number, e-mail address, address etc) within a given time frame, then most likely the user was a customer of that hotel and the review is trustworthy. Otherwise, there is no guarantee that that is the case.