COMP1204 - Data Management
UNIX Coursework Report

Giovanni Arcudi

MEng - Computer Science with Artifcial Intelligence
UNIVERSITY OF SOUTHAMPTON
-
Student ID: 31275206
E-mail: ga1g19@soton.ac.uk

April 3, 2020

Contents

1 Scripts

In this section will be discussed all the versions of the `countreviews.sh` Bash script I developed to analyse the TripAdvisor's dataset and find out what hotels have more reviews.

1.1 Premise

Before starting to write the script, I analysed the dataset in order to identify the structure and the pattern of each `.dat` hotel file, and I noticed that everyone had:

- One `<Overall Rating>` tag, storing the overall rating of the hotel.
- One `<Avg.Price>` tag, storing the average price per night, in dollars, of the hotel.
- One `<URL>` tag, storing the link to the hotel Tripadvisor's webpage.
- Many reviews, internally containing:

    1. One `<Author>` tag, storing the nickname of the author.
    2. One `<Content>` tag, storing the review's text.
    3. One `<Date>` tag, storing the date the review was published.
    4. One `<No.Reader>` tag, storing the number of readers.
    5. One `<No.Helpful>` tag, storing the number of users that find the review helpful.
    6. One `<Overall>` tag, storing the overall rating given by the user.
    7. One `<Value>` tag, presumably storing a "value for money" rating.
    8. One `<Rooms>` tag, storing the rating related to the quality of the rooms.
    9. One `<Location>` tag, storing the rating related to the quality of the hotel location.
    10. One `<Cleanliness>` tag, storing the rating related to the cleanliness of the hotel.
    11. One `<Check in / front desk>` tag, storing the rating related to the quality of the front desk staff.
    12. One `<Service>` tag, storing the rating related to the quality of the service.
    13. One `<Business service>` tag, storing the rating related to the quality of the business service.

1.2 Version 1 - Single hotel reviews counting

As per instructions, the first version of my *countreviews.sh* bash script counted the number of reviews that a hotel has received, reading from its `.dat` file. The most reasonable way to do that was to count the number of occurrences of the `<Author>` tag.

The source code for the first version of the script, named *countreviews.sh*, is shown below:

```
1 #!/bin/bash
2
3 fileName = "$1"
4
5 grep -c "<Author>"  $fileName
```

Script 1: Version 1 of *countreviews.sh*.

Code analysis

The first line represents the *shebang*, which tells the interpreter to execute the file by using the Bash Shell.

In the third line, there is a variable called `fileName`[1], in which the name of the `.dat` file is stored. The name of the file is specified as first parameter by the user when running the script, therefore it is referenced as `"$1"`.

Finally, in line 5, I use the `grep` command with the `-c` option, which returns the number of occurrences of the string `"<Author>"` within the `.dat` file.

An example of Script 1's functionality is shown in Figure 1 below.

```
% ./countreviews.sh <hotel_fileName>.dat
15
```

Figure 1: Usage and Output of *countreviews.sh* (version 1).

[1]Even though the usage of this variable was not required for the functionality of the script, because it is only accessed once and so does not heavily influence performance, I decided to use it in order to make the code more readable.

1.3 Version 2 - Multiple hotels reviews counting

For my second version of *countreviews.sh*, as requested by the coursework instructions, I modified the first version of the script to count the number of reviews in each hotel file given the (absolute or relative) path to the folder containing all the `.dat` files.

The source code for the second version of the script, named *countreviews.sh*, is shown below:

```
1 #!/bin/bash
2
3 reviewsFolderPath="$1"
4
5 for i in $reviewsFolderPath/*
6   do grep -c "<Author>" $i
7 done
```

Script 2: Version 2 of *countreviews.sh*.

Code analysis

As in the previous version of the script, the first line represents the *shebang*, which tells the interpreter to execute the file by using the Bash Shell.

In the third line, there is a variable called `reviewsFolderPath`[2] , in which the (absolute or relative) path to the folder containing all the hotels `.dat` files is stored. The path is specified as first parameter of the script when the user runs it, therefore it is referenced as `"$1"`.

In line 5, a for loop that iterates over each file in the hotels' files folder is initialised. In particular, `i` represents a generic file and the expression `$reviewsFolderPath/*` represents all the files in that directory.

In line 6, I use the `grep` command with the `-c` option to print out the number of occurrences of the string `"<Author>"` within the file `i`; which, as stated in the previous page, is equivalent to count the number of reviews.

Finally, in line 7, the loop is closed.

An example of Script 2's functionality is shown in Figure 2 below.

```
% ./countreviews.sh <path_to_reviews_folder>
15
12
13
...
```

Figure 2: Usage and Output of *countreviews.sh* (version 2).

At the time of the implementation, I was aware of the fact that using a for loop would have imperceptibly slowed down the speed of the script but, because the script had to output just numbers, I thought that this implementation, despite imperceptibly slower, would have been a lot more clear.

Of course, for my final *countreviews.sh* implementation (whose output has to contain also the hotel's name), I applied all the modifications necessary to, even if imperceptibly, speed up performance.

[2]Even though the usage of this variable was not required for the functionality of the script, because it is only accessed once and so does not heavily influence performance, I decided to use it in order to make the code more readable.

1.4 Version 3 - Multiple hotels reviews counting and ranking

For the final version of *countreviews.sh*, I:

- Modified the script in order to output a descending ranking of all the hotels according to the number of reviews (formatted as stated in the Coursework's specifications).
- Maximized the performance of the script by removing the for loop (which was convenient to use in the previous version because the script did not have to output the name of the hotel) and instead taking advantage of the `grep` command's "built-in" loop functionality.

The source code for the third and last version of the script, named *countreviews.sh*, is shown below:

```
1 #!/bin/bash
2
3 reviewsFolderPath="$1"
4
5 cd $reviewsFolderPath
6
7 grep -c "<Author>" * | sed "s/\.dat:/ /g" | sort -k 2 -nr
```

Script 3: Final version of countreviews.sh

Code analysis

As in the previous versions of the script, the first line represents the *shebang*, which tells the interpreter to execute the file by using the Bash Shell.

In the third line, there is a variable called `reviewsFolderPath`[3] , in which the (absolute or relative) path to the folder containing all the hotels' `.dat` files is stored. The path is specified as first parameter of the script when the user runs it, therefore it is referenced as `"$1"`.

At line 5, we find the first "proper" command of the script, `cd` (change directory), which allows to go inside the directory, either using an absolute or relative path. By working inside the directory, the script avoids iterating over each file, as the `grep` command will do that for us.

[3]Even though the usage of this variable was not required for the functionality of the script, because it is only accessed once and so does not heavily influence performance, I decided to use it in order to make the code more readable.

At line 7, we find the pipe responsible for performing our target operation. By just executing `grep -c "<Author>" *`, which returns the number of occurrences of the string "<Author>" within each file of the directory, we already obtain most of what we are looking for (see Figure 3).

```
hotel_100504.dat:125
hotel_100505.dat:163
hotel_100506.dat:33
...
```

Figure 3: Output of `grep -c "<Author>" *`.

However, the desired output is the name of the hotel and the number of reviews, separated by a single space. Therefore, to obtain that, `grep`'s output is piped into the `sed` command, which will remove the substring "`.dat:`" and will replace it with a single space.

Lastly, the just obtained output is piped into the `sort` command, which will sort the number of reviews in descending order, thanks to the following 3 options:

- `-k2`: sort the second column.
- `-n`: sort according to string numerical value.
- `-r`: reverse order, since the sort command outputs by default in an ascending order.

The final usage and output of the script is shown in Figure 4 below.

```
% ./countreviews.sh <path_to_reviews_folder>
hotel_218524 2686
hotel_149399 1551
hotel_208454 1223
...
```

Figure 4: Usage and Output of *countreviews.sh* (version 3).