DEVELOPMENT OF A PYTHON PROGRAM TO STUDY THE CHANGE IN CHARACTER OCCURRENCES IN THE TELEVISION SHOW GAME OF THRONES

May. 1, 2017

RUYI LYU

1682602

INTRODUCTION

The most significant skill that a computer scientist has to master is probably problem-solving. [1] Having features such as while loop, computer languages are particularly good at repetitive tasks like statistical analysis. Moreover, as a high-level language, Python is designed to be easy for the human to understand and thus economical to use for simple studies. It allows programmers to solve problems in fewer lines compared to C++ or Java. [2] Python is also known for its extensibility, not requiring all the functions and methods to be built in the language core. [3] A compact and flexible program can thus be developed.

In this study, a Python program is developed to extract, save and process information from a website called 'Game of Thrones Wiki'. The hyper-textual network contains a great quantity of data like episode plots, character introductions, world cultures centering the television show 'Game of Thrones', which is known for its enormous and complex character network. This study makes use of a series of links that contain episode plot introduction. The occurrences of a character are important indicators of the importance of the character during certain periods of the show. It is thus interesting to extract this information from the website and compare the change in character occurrences as visually as possible.

MATERIALS AND METHODS

Useful Python Functions and Tools: The main structure of the program is a while loop that compels the body program to run again and again as long as the visited page number has not yet reached the maximum page number stipulated. In the program submitted the maximum page number was set to be 60. This is because only 60 episodes have been aired before the submission. This number can be changed easily in case more episodes were aired or the user wanted to study certain period of the show instead of the full length.

```
while page <=max_pages:
```

Figure 2.1 While loop in the program

Three functions are defined in the program: the fundamental function GOT_spider; the character counting function count_cha; the plotting function ribbon_plot. Each time the program goes through the while loop, count_cha returns a tuple that contains the occurrences of characters in the visiting episode. The GOT_spider then saves the data in the form of a list into a list called 'datall'. This list will later be used as an argument of ribbon_plot to produce ribbon plot of character occurrences. It is worth pointing out that output and ribbon plot are implemented only after all the counting is done.

```
def GOT_spider(max_pages, startpg):

def count_cha(soupdata):

def ribbon_plot (datalist):
```

Figure 2.2 Three functions in the program

Logging and Output File: Two new files will be generated after running the program: the log file 'GOTspider.log' and the output file 'GOToutput.json'. The log file takes note of two events showing the ongoing process: visiting episode website and counted character occurrences. The format of messages was set to be %(asctime)s %(levelname)s %(message)s and the minimum level was set to be 'INFO' to allow all the information to pass to the log file. The output file shows the raw data that is used to build up the ribbon plot. This allows the user to study the special occasions and tendency more meticulously. The keys of the dictionary in the output file are the episode numbers and the occurrences are arranged in the same order as in the ribbon plot. It should be noticed that the log file is written as the program progresses while the output file is only generated at the last step of the function GOT_spider.

```
"5": [
    29,
    6,
    6,
    8,
    10,
    16,
    4,
    1,
    0,
    0
595 2017-05-02 10:32:41,959 INFO Visiting episode 55 http://gameofthrones.wikia.com/wiki/The_Door
596 2017-05-02 10:32:42,077 INFO Robert has appeared: 0 time(s)
597 2017-05-02 10:32:42,077 INFO Jon has appeared: 6 time(s)
598 2017-05-02 10:32:42,077 INFO Arya has appeared: 8 time(s)
599 2017-05-02 10:32:42,078 INFO Cersei has appeared: 1 time(s)
600 2017-05-02 10:32:42,078 INFO Jaime has appeared: 0 time(s)
601 2017-05-02 10:32:42,078 INFO Tyrion has appeared: 7 time(s)
602 2017-05-02 10:32:42,079 INFO Daenerys has appeared: 9 time(s)
603 2017-05-02 10:32:42,079 INFO Petyr has appeared: 1 time(s)
604 2017-05-02 10:32:42,079 INFO Jaqen has appeared: 4 time(s)
605 2017-05-02 10:32:42,080 INFO Hodor has appeared: 18 time(s)
```

Figure 2.3 Samples of the output file and the log file

Information Selection and Data Pattern: Even in a single episode page in the game of thrones wiki website, there are a large number of links that lead to character pages, other episode pages, location pages and etc. To extract the link to the next episode of the episode the program is currently visiting, a certain pattern of the 'next episode links' needs to be found. It was discovered that the link to the next episode is always in a table (and the only table in the page) and it is always the last link in the table. Therefore, the following codes were written to first locate the table in the page, then saves all links into a list 'epillist' and finally extract the last link in the list.

```
for table in soup.findAll('table', {'class': 'pi-horizontal-group'}):
    epilist = []
    for link in table.findAll('a'):
        epilist.append(str (link.get('href')))

next_epi = epilist[len(epilist)-1]

page += 1
```

Figure 2.4 Extract next episode link

All the words in the plot introduction can be located by finding the 'p' tap in the soup object. Thus simply counting the character name in the parts that are tagged as 'p' we can obtain the occurrences in a single episode.

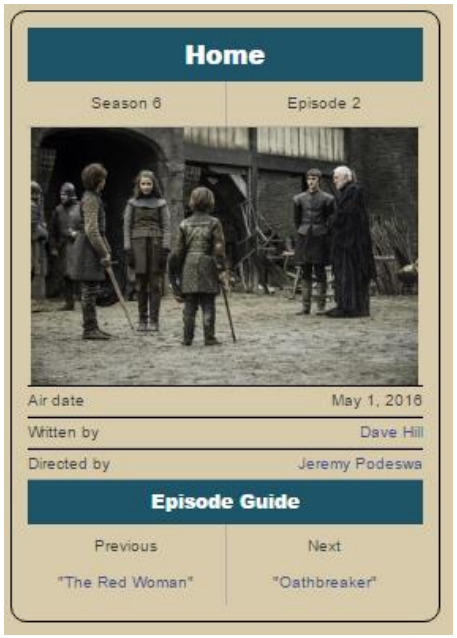


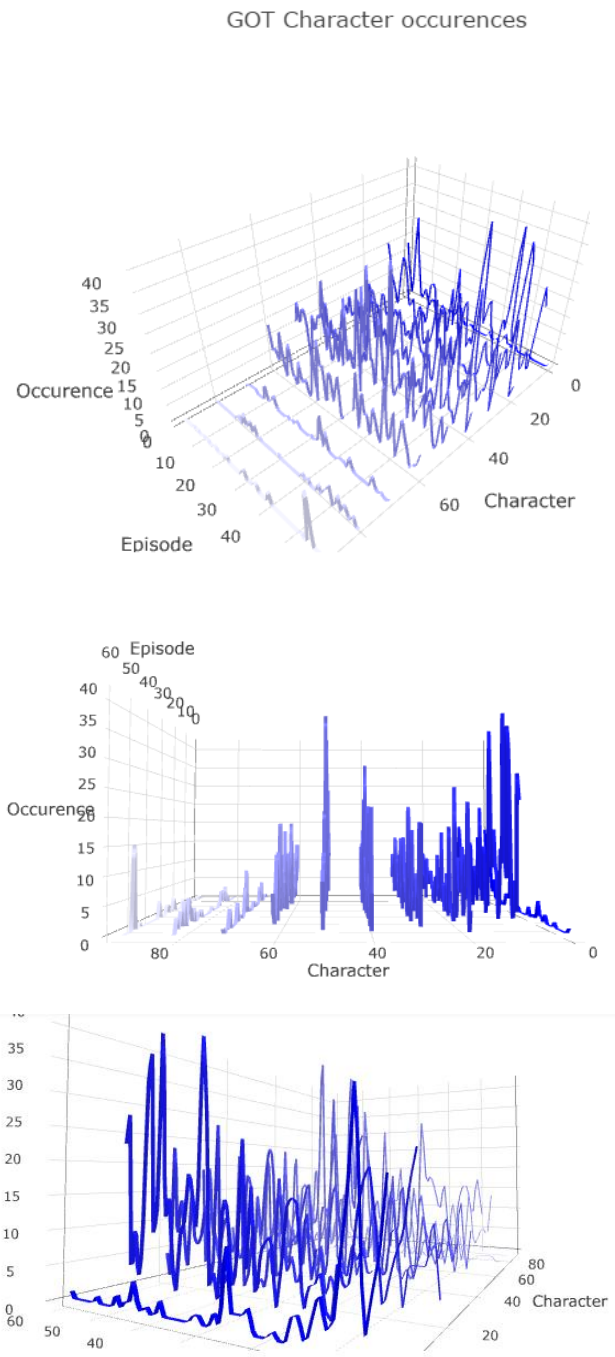
Figure 2.5 Keywords in one episode of Game of Thrones (Retrieved from http://gameofthrones.wikia.com/wiki/Home)

Data analysis: If the researcher wants to verify a hypothesis related to the character occurrences, statistical tests like T-test must be conducted. However, in this study, only the general tendency is the study object, thus all the data was used to obtain a ribbon plot in order to view them directly. The z value is the calculated occurrences, the y value is the episode number and the x value if just for separating the 10 traces.

RESULTS

All raw data can be found in the output file in the format already explained. It will thus not be attached here. The ribbon plot obtained is attached here from different perspectives.

```
Character list:
Trace 0: Robert Baratheon
Trace 1: Jon Snow
Trace 2: Arya Stark
Trace 3: Cersei Lannister
Trace 4: Jaime Lannister
Trace 5: Tyrion Lannister
Trace 6: Daenerys Targaryen
Trace 7: Petyr Baelish
Trace 8: Jaqen H'ghar
Trace 9: Hodor
```



DISCUSSION

A. Character Importance Tendency

It can be seen that the importance of characters can be classified into at least three categories based on occurrence tendency. The first type is 'decaying importance' like Robert Baratheon. This makes sense because Robert Baratheon, though the king of seven kingdoms, died in the first season. The second type is 'stabilized importance' like Jon Snow, Cersei Lannister, and Daenerys Targaryen. These characters continue to bloom despite others' miserable fates. The last type would be 'limited importance' like Petyr Baelish, Jaqen H' ghar and Hodor. These characters, though sometimes essential for the development of the plots, are usually not treated as main characters.

B. Error Analysis

Although the results fit the general sets of the television show, some possible errors were noticed and investigated. For instance, Robert Baratheon continued to appear in the plot despite his death in season one. After checking the original websites, it was confirmed that this is either due to other characters' mention of the king or telling a past event that involves the king. However, in the case of Jon Snow, some solid errors do occur. Only the characters' first names were used in the search because they do not always appear in full names. Thus the count of Jon Snow, though with trivial effect, also included the count of Jon Arryn in the plot. Furthermore, some picture titles also include character names. These counts should not really be included because the same plots have already been covered in the word parts.

The first type of error can be perfectly solved only if all the character names are linked to the character pages, which is not the case in Game of Thrones Wiki. The solution to the second type of error requires identification of picture titles. This could probably be completed by identifying the fonts.

C. Limitation and Further Development

As already mentioned in the previous part, the ribbon plot obtained can only be used as evidence of general tendency. If one needs to verify a hypothesis, for instance, whether the speed of increase in occurrences of Jon Snow is larger than the speed of decrease in occurrences of Robert Baratheon, further statistical tests have to be conducted to draw any conclusion.

D. Network Selection

The network Game of Thrones Wiki was selected for its wide coverage and mass information quantity. In this sense, the network has successfully provided enough nodes and links to process. Furthermore, the large sample size renders the tendency more obvious and accurate.

However, the network is only a recreational site created by fans. Thus the uneven distribution of word count in different plots, the predilection of certain characters could all add to the uncertainty of this

study. Furthermore, as mentioned previously, the lack of a link on some of the character names decreases the accuracy of character counts. Nevertheless, overall, the network selected has fulfilled the basic requirements of the study.

CONCLUSION

To a certain extent, the Python program developed could help us study the change in character occurrences in the television show Game of Thrones. Three types of importance were classified: 'decaying importance' 'stabilized importance' and 'limited importance'. Errors regarding character counts still occur from time to time and statistical tests are required if hypothesis needs to be verified. The network selected has fulfilled the basic requirement and overall, the program works efficiently.

REFERENCES

[1] Nantais, C.L. and Downey, A.B., 2013. How to Think Like a Computer Scientist.

[2] Summerfield, M., 2007. *Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming*. Pearson Education.

[3] Venners, Bill. 2003. *"The Making of Python"*. Artima Developer. Artima