The Dream of Red Mansions- Who loves smiling more?

Yingying Han

I. Downloading texts and validation process

Originally, I planned to download both Chinese and English version from http://www.gutenberg.org because Gutenberg website. Luckily, both versions are found from: http://www.gutenberg.org/ebooks/search/?query=Hong+Lou+meng.

English version:
Book name: The dream of the red chamber: Hung lou meng, books l and ll
Translator: H Bencraft Joly
Book information on World Cat: https://www.worldcat.org/title/dream-of-the-red-chamber-hung-lou-meng-books-l-and-ll/oclc/795566543?referer=di&ht=edition
Book I downloaded from: http://www.gutenberg.org/ebooks/9603
Book II downloaded from: http://www.gutenberg.org/ebooks/9604

Chinese version:
Book name: Hong Loumeng
Author: Cao Xueqin
Downloaded from: http://www.gutenberg.org/files/24264/24264-0.txt

II. Validation of the downloaded text:

Since there are 120 chapters in the original Chinese book, as shown in the downloaded Chinese version. But there are only 56 chapters found in the English version (BookI+BookII). By comparing, I found Chapter LVI in English version contains the same contents as in Chapter LVI in the Chinese version. Later, I figure out there is a note by the translator at the end of *The dream of the red chamber: Hung lou meng, Book ll*:
"The second volume of this translation ends thus,and no more of it was ever published."

Though feeling strange about why the translator only translated only 56 chapters, I could not work on this English version, because it is not an equivalent of the Chinese version. Two solutions were put forward at this step:
(1) Analyze only the first 56 chapters of the Chinese version too;
(2) Downloading another authorized English version if available.

Further searching from website (https://site.douban.com/bookschina/widget/notes/1969803/note/216115486/)figures out that there are two completed English version of this book. One is the version translated by Xianyi Yang and Gladys Yang and the other is the version translated by David Hawkes.

However, searching on "http://www.gutenberg.org/" , "https://archive.org/", I fail to get access to the txt file of both English books.

I have to take the first solution I put forward: analyzing only the first 56 chapters of the book.

While reading the file, I found my PyCharm could not read the whole English novel file. Professor Bosch suggests me to read the file as binary and then decoding it. The code is as followed:

```
   infile=open("EnglishVer.txt","rb")
   allthelines=infile.read().decode(errors='replace')
```

But after a few tries, I still failed to read the whole file.

I found *The dream of the red chamber: Hung lou meng, Book l* include Chapter 1 to Chapter 24 and *The dream of the red chamber: Hung lou meng, Book ll* include Chapter 25 to Chapter 56. PyCharm could read Book I while failed to read Book II. **Thus, I finally decide to analyze the first 24 chapters of this book.**

III. English File Analysis

1. Text pre-processing of English file

Text pre-processing includes only text normalization. To be specific, text normalization includes:

(1) Extract the first 24 chapters contents from the whole English file

Narrative:

a. I use the "find" function to get the index where the chapters start and end.
b. Then I slice the string from the start index to the end index: chapters = allthelines[startpoint:endpoint]
c. By doing so, I extract the first 24 chapters out from the English file.

(2) Convert text to lowercase

Narrative:

I use the "lower" function to lowercase the chapters contents: lower_chapters = chapters.lower()

(3) Remove punctuation

Narrative:

a. I import string, and get all the punctuations in the string

b. I use a for loop, and replace function. Whenever there is a punctuation, I will replace that with a whitespace.

(4) Remove numbers

Narratives:
a) To remove numbers, I define a function. The function name is “removeNumberFromStrings” and parameter is a string.

b) I create an empty string. The reason why it is an empty string instead of an empty list is that I open the file with .read() instead of .readlines(). So, I want the file with no numbers could be still read as a string instead of a list.

c) For each character in that string, if that character is a digit (0-9), then do not add that character to the empty string (newString = newString). Otherwise, attach that word to the empty string.

d) I call the function and print out the return results to see if this function works well.

(5) Remove whitespaces

Narrative:
I use .strip() function to remove the leading and trailing white spaces in the text.

(6) Explore the data: the most frequent appeared 200 words

Narratives:
a) Until now, all I have is a string. But latter, I want to deal with “word” instead of “letter”. So, I need to use .split() function to transfer a string to a list, in which, each word of the file is an element.

b) What need to be point out is that it does not shown all the elements in the list out in the shell. I searched online about the reason, I think it is because the setting of PyCharm and you could only maximumly see 400 elements one time.

c) I want to have a dictionary, in which, the key is the word appears in the English file and the value is the number the word occurs.

d) Thus, I create a new empty dictionary named counts.

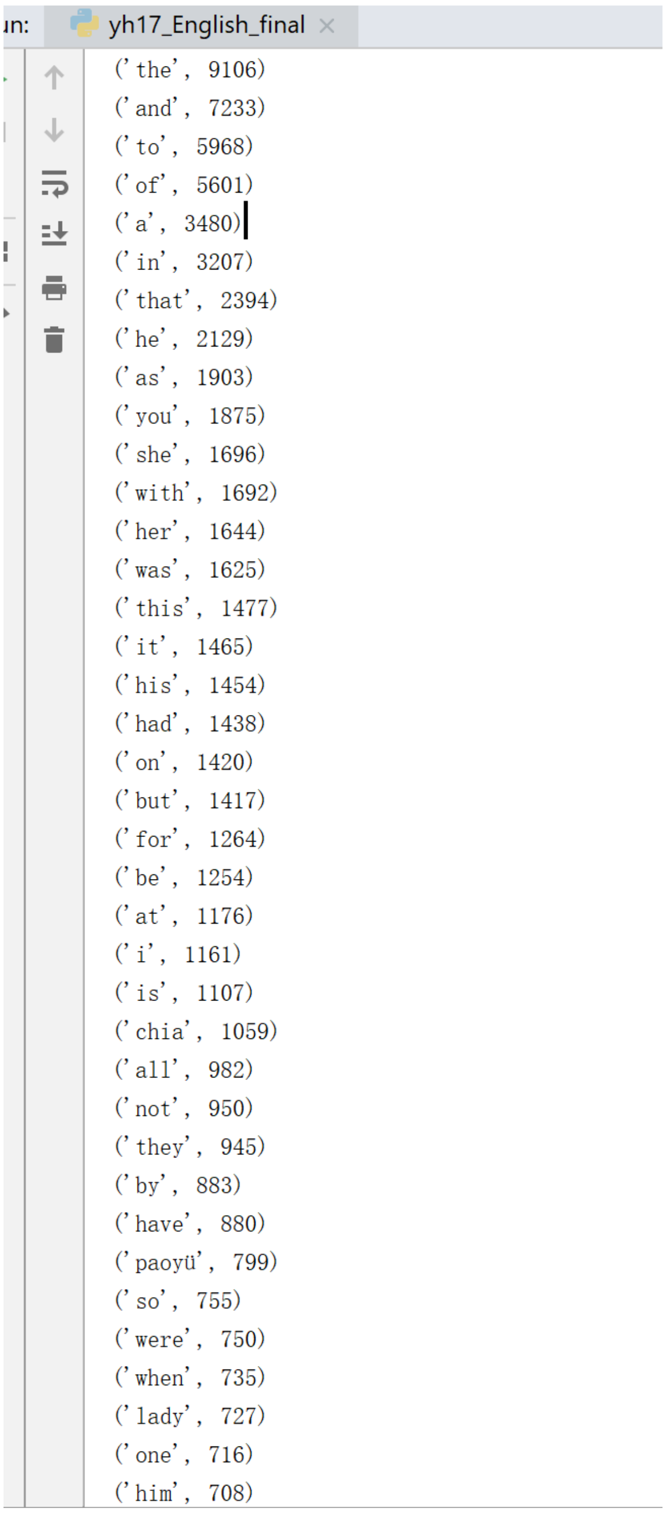
e) For each word in the file, if it is the first time it appears in the file, then it counts 1 time. Otherwise, counts = counts+1

f) Then I put all the dictionary into items, with a pair of data value in an item.

g) For each pair in the item, the “word” is the key, and the “counts” is the value.

h) I defined a new function to sort the results based on the counts each word occurs.

i) I want to see the most frequent 200 words. Part of the results are shown in Figure 1. As shown, it is necessary to omit stop words from my counts.



```
('the', 9106)
('and', 7233)
('to', 5968)
('of', 5601)
('a', 3480)
('in', 3207)
('that', 2394)
('he', 2129)
('as', 1903)
('you', 1875)
('she', 1696)
('with', 1692)
('her', 1644)
('was', 1625)
('this', 1477)
('it', 1465)
('his', 1454)
('had', 1438)
('on', 1420)
('but', 1417)
('for', 1264)
('be', 1254)
('at', 1176)
('i', 1161)
('is', 1107)
('chia', 1059)
('all', 982)
('not', 950)
('they', 945)
('by', 883)
('have', 880)
('paoyü', 799)
('so', 755)
('were', 750)
('when', 735)
('lady', 727)
('one', 716)
('him', 708)
```

(7) Remove stop-words

Narratives:
a.) I need to read the stop words text file through readlines()
b.) I want to remove the white spaces after each word, so I use .strip() function here.

c) I create an empty list and store each stop word without white space into the list.
d) Then for each word in the novel, if it is not in the stop word list, then append that word to a new list. By doing so, I could remove all the stop words.
e) Like what I did before, I use counter to see the most frequently occurred 200 words.

It is necessary to point out I remove the step6 and step 7 codes out from the final version because it is not directly helpful for data analysis. I store the codes in a file named "yh17_English_preprocessing". The reason why I still keep and submit these codes is that it is necessary to see if the names of the three main characters and the word "smile" occurs enough time.

(8) Pre-analysis before extracting valued data
The name of three main characters and word "smile" all occur many times in the first 23 chapters. According to the results, "Pao-yü" occurs 799 times. Tai-yü occurs 246 times, and Paochia occurs 76 times. the word "smile" occurs 116 times and "smiled" occurs 81 times.

2. Extract the valued data

Since this project focus on who, among the three main characters, smiles more. But the punctuations have been removed from the original text at this stage. I decide to have a new python file (yh17_Englishfile.py) to extract the valued data.

(1)
I firstly extract all the sentence with "smil", and explore the sentences:
Narrative:
a) I split the sentences by ",";

b) I created an empty list named "smile_sentences";

c) For each sentence in the file, if the string "smil" is in, then, extract this sentence.

d) Finally, I have a list with all sentences with string "smil" in. There are 320 elements.

(2)
Then I want to see how many time Pao-yu, tai-yu, bao-chia smiles:

Narrative:

I created a new empty list, for each sentence in the smile sentences extracted from last step, if it includes string "Pao-yu smil", then extract that sentence out and store that to the empty list.

Using the same loop method, I find pao-yü, tai-yü, and paochia smiles 17, 4 and 0 times respectively, which is lower than I expected.

(3)
Now I start to look back at the sentences with “smil”. There are many sentences like “he smiles”, “he said with a smile”. In this case, Pycharm will fail to identify who do “he”, “she”, “they” mean. Also, there might be sentences like “Pao-yu talk to the girls for a while and then he came back with a smile”. In this case, the word “smile” is not after word “pao-yu”, but exists in the same sentence with the word “pao-yu”.

Also, the character name might exist in different ways, which is shown in the following table:

Official name	**Possible names in the text**
pao-yü	pao yü pao-yü
tai-yü	tai-yü
paochia	pao chien pao-ch'ai lady chai

I decide to change the way I split the text. Instead, I try to split it with period. Also, I am not going to detect how many time “smil” occurs after the occurrence of the character name, but how many times the string “smil” appear in the same sentence with the characters’ names.

Narrative:
Using a for loop and an if statement, if “Paoyü” is in each sentence and “smil” is in each sentence, then extract the sentence out.

Using the same methods for three times, I find Paoyü, tai-yü, and paochia smile 94 times, 40 times and 13 times respectively.

Finally, I write the sentences out to three files.

IV. Chinese File Analysis
1. Data pre-processing
(1) I need to read the file.

Narratives:
a) Though I have saved the text file with encoding “utf-8”, I still have problem with opening the file. Search online, I find it works by add a new line # -*- coding: UTF-8 -*-.

b) I read the whole file with .read(), and thus, the file is read in a string.

(2) Extract the first 24 chapters contents from the whole English file
I adopt the same method as pre-process English file. By finding the index of the starting word and ending word. I extract the first 24 chapters out.
It is necessary to point out that there is no “lowercase” and “number” in the Chinese file. Thus, for next step, I am going to remove the white spaces.

(3) Remove whitespaces
I use .strip() function to remove all the whitespaces at the beginning and end of the text.

(4) Chinse word segmentation
Before removing stop words, I need to segment the text. In English file, the character name might be written like “pao yü” and the stop words will not include the word “pao” or “yü”. However, in Chinese, the stop words might include the Chinese character of the names. So, I need to segments the words, thus, ParCharm could understand “寶玉” is a phrase and are not two separate characters.

Narrative:I install and import Python Chinese word segmentation module: jieba.

5) Remove stop words and counts

Narrative: I download Chinese stop words list from the Internet. Adopting the same method as in English anlysis file, I remove stop words and figure out the most frequently appear words.

The three main characters’ name “寶玉” (Pao-yü),”黛玉” (Tai-yü), and “寶釵”(Paochia) appears 401 times, 94 times, and 65 times respectively. And “笑” appears 373 times.

2. Extract valued data
(1) Split the file by period.
Narrative:
a) The file I have now is a list. I need to use for loop to transfer it to a string.
b) Then I split it by “。”

(2) Extract valued data
Narrative:

Adopting the same method, I find Paoyü, tai-yü, and paochia smile 213 times, 88 times and 44 times respectively.

V. Conclusion

The result is shown in Table 1. Paoyü, among the three characters, loves smiling most in both Chinese novel and English novel. Any paoyü, tai-yü, paochia smile more times in Chinese novel than in English novel.

Table 1: analysis result

Character name	**Text Language**	**Smile times**
paoyü	English	94
	Chinese	213
tai-yü	English	40
	Chinese	88
paochia	English	13
	Chinese	44