FIT 9133 Assignment 3

Tan Kah Wang

29442826

Table of Contents

1. **Task 1: Setting up the preprocessor** 2
2. **Task 2: Building a class for analysing characters**........................2
3. **Task 3: Building a class for analysing words** 2
4. **Task 4: Building a class for visualising the analysis**....................2
5. **Task 5: Putting all the classes together** 3
6. **Program Description**...3
7. **Assumptions** 4

1. Task 1: Setting up the preprocessor

In this task, we are supposed to define a class that will perform the basic pre-processing on the 6 input texts. An instance variable *token_list* has been defined which will be returned in the method *get_tokenised_list* which will be required for further analysis in Task 2 to Task 5. __*str*__ method was defined to return the total number of tokens for the tokenised input text in the form: "Total number of tokens: *count ".* The *tokenise* method was defined to split the given input text into individual tokens and in this method, all alphabets were converted to upper case for ease of analysis for the remaining tasks.

2. Task 2: Building a class for analysing characters

In this task, we are supposed to define a class for analysing the number of occurrences for each character from the 6 tokenised list obtained from task 1. An instance variable *character_count* has been defined as a data frame from Pandas package. __*str*__ method was defined to return the count of each character in the form: "*character* has count of *count*". The *analyse_characters* will first join the tokenised_list from task 1 as a string and count for each character their count and add to the instance variable *character_count*. It is then sorted alphabetically and returned in this method. The *get_punctuation_frequency* method will count punctuations (i.e. characters other than 'A to Z' and '0 to 9') from the instance variable *character_count* and return as a Pandas data frame *punctuation_count*.

3. Task 3: Building a class for analysing words

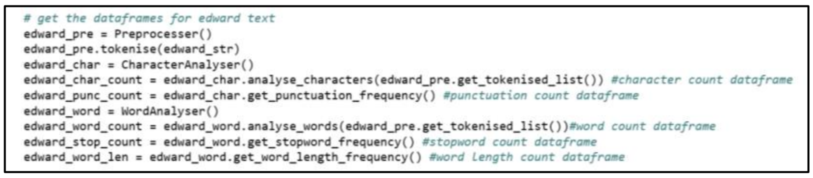
In this task, we are required to define a class for analysing the number of occurrences for each word from the 6 tokenised list obtained from task 1. An instance variable *word_count* has been defined as a Pandas data frame to store the counts of the words. __*str*__ method was defined to present the words occurrences in the format: *"word* has count of *count*". The *analyse_words* method accept the tokenised list as an argument and attempt to count the occurrences of each word and append to the instance variable *word_count* which will be returned in this method. The next method in this class *get_stopword_frequency* will return only the frequency of stopwords based on the list obtained from Onix Text Retrieval Toolkit. A request was made to the url of the stopwords list and the stopwords list for comparison was obtained by slicing the data list from the word "*about"* to "*yours".* Note that all single character in the stopwords list has been removed. This list was then compared to the instance variable *word_count* and will only return stopwords in another Pandas data frame *stopword_list_count* which will be returned at this method. The last method in this class *get_word_length_frequency* was defined to return the counts of word length of all words found in the 6 input texts. A Pandas data frame *word_length_count* was created to store the word length counts and returned at the end of this method. Note that a prefix "*Word Length"* has been added to this data frame for each count (e.g. "*Word Length: 1"*) so as to avoid confusion with numeral character counts in Task 2's *analyse_characters* method.

4. Task 4: Building a class for visualising the analysis

In this task, we are required to define a class for visualising the stylometric analysis. An argument *all_text_stats* must be involved when calling this class. With the argument *all_text_stats* which contains the statistics of the 6 input texts (obtained from Task 5), the __*init*__ method of this class will get the relative frequencies for character, punctuation, stopword and word length and append them into a Pandas data frame *overall_df_rel* which is the instance variable in this class. Note that the relative frequency of punctuation was not calculated separately as punctuations has already been included while calculating character frequency. The next method *visualise_character_frequency* will return a bar chart (refer to *charplot* file attached in this submission) of alphabets and numerals against their relative frequencies which can be obtained from the instance variable *overall_df_rel*. Similarly, the next method *visualise_punctuation_frequency* will return the relative frequencies of punctuations found in the 6 texts which can be obtained from the instance variable *overall_df_rel* as a bar chart (refer to *puncplot* file attached in this submission). Next method *visualise_stopword_frequency* will return the relative frequencies of stopwords found in the 6 texts as a bar chart (refer to *stopplot* file attached in this submission). The next method in this class *visualise_word_length_frequency* will return the relative frequencies of word length for the 6 input texts as a horizontal bar chart (refer to *lenplot* file attached in this submission). An additional method *return_relative_frequency* was defined to return the instance variable.

5. Task 5: Putting all the classes together

In this task, we are required to put all the classes together as a program to analyse the 6 provided input texts. A *read_input()* function was defined for reading in each of the 6 input texts and potential I/O exceptions which include *FileNotFoundError, IOError & RuntimeError* were considered. The *main()* function will then read in the 6 input text files using the *read_input*() function and the 3 classes in Task 1, 2 and 3 will be called upon to create objects pertaining to their tasks. A total of 5 data frames will be obtained for each input text, mainly character count, punctuation count (from Task 2), word count, stopword count and word length count (from Task 3) shown in Figure A.



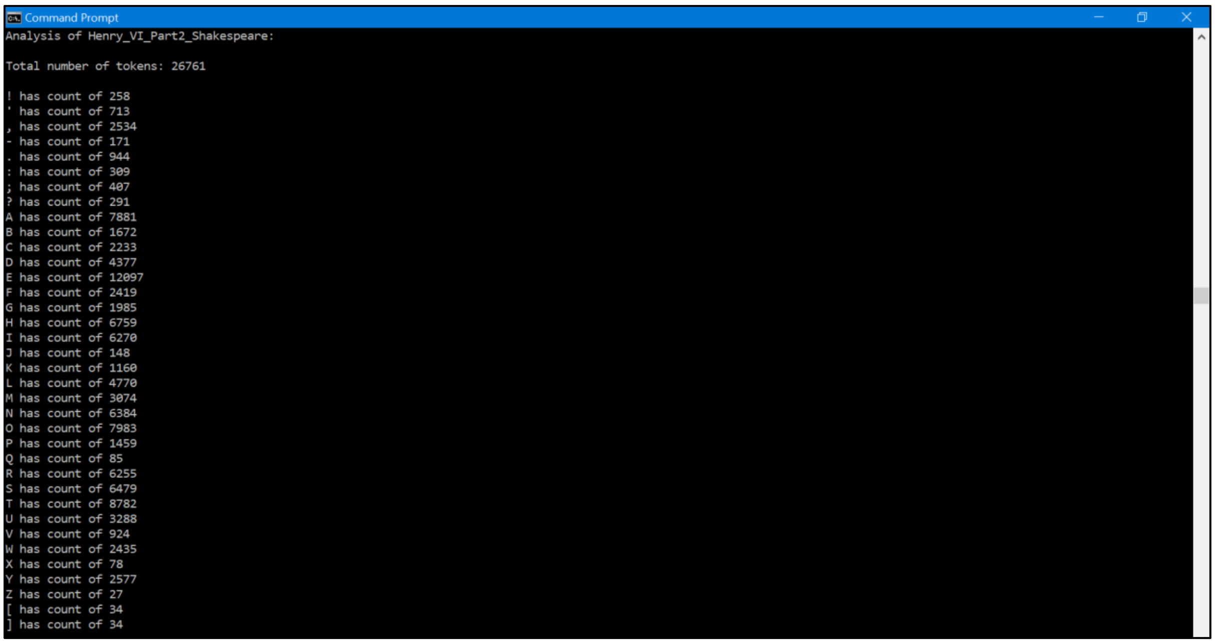
```
# get the dataframes for edward text
edward_pre = Preprocesser()
edward_pre.tokenise(edward_str)
edward_char = CharacterAnalyser()
edward_char_count = edward_char.analyse_characters(edward_pre.get_tokenised_list()) #character count dataframe
edward_punc_count = edward_char.get_punctuation_frequency() #punctuation count dataframe
edward_word = WordAnalyser()
edward_word_count = edward_word.analyse_words(edward_pre.get_tokenised_list())#word count dataframe
edward_stop_count = edward_word.get_stopword_frequency() #stopword count dataframe
edward_word_len = edward_word.get_word_length_frequency() #word length count dataframe
```

(Figure A – Creating objects and instances in Task 5 by calling upon Task 1, 2 & 3)

This process will be repeated for the 6 input texts and at the end, 18 data frames will be combined into one as *my_all_text_stats* which contains the overall statistics of the 6 input texts, which is also the argument needed to create an object for Task 4. Note that, punctuation data frames were not included in *my_all_text_stats* as punctuations have already been included in the characters counts. After creating an object based on Task 4, methods defined in class of Task 4 were called upon to create plots for characters, punctuations, stopwords and word lengths. The overall statistics of relative frequencies is also returned as a file (refer to *relativefrequency* file attached in this submission) by calling upon *return_relative_frequency* method from Task 4.

6. Program Description

When the *main_29442826.py* is run, the program will output the 4 plots as 4 different files (i.e. charplot, puncplot, stopplot, lenplot) which contains the bar charts of relative frequencies of characters, punctuations, stop words and word lengths for the 6 input texts respectively. Furthemore, the 3 __*str*__ methods that were created in Task 1, 2 and 3 will be printed on the terminal as shown in Figure B for each of the 6 input texts.



```
Analysis of Henry_VI_Part2_Shakespeare:

Total number of tokens: 26761

! has count of 258
' has count of 713
, has count of 2534
- has count of 171
. has count of 944
: has count of 309
; has count of 407
? has count of 291
A has count of 7881
B has count of 1672
C has count of 2233
D has count of 4377
E has count of 12097
F has count of 2419
G has count of 1985
H has count of 6759
I has count of 6270
J has count of 148
K has count of 1160
L has count of 4770
M has count of 3074
N has count of 6384
O has count of 7983
P has count of 1459
Q has count of 85
R has count of 6255
S has count of 6479
T has count of 8782
U has count of 3288
V has count of 924
W has count of 2435
X has count of 78
Y has count of 2577
Z has count of 27
[ has count of 34
] has count of 34
```

(Figure B – Command Prompt output after running main_29442826)

Lastly, the statistics for the 6 input texts which contains their relative frequencies for characters, punctuations, stop words and word lengths was created as a Microsoft Excel csv file named *relativefrequency* which has been included in this submission.

7. Assumptions

The first assumption made in this assignment was alphabets in the 6 input texts were not case sensitive, all alphabets in the texts have been converted into upper case in this assignment.

The second assumption made was stop words consisting of only one single character has been removed from stop word list and were not considered as stop word.

The third assumption made was since it was stated in the assignment description that all input text files were in tokenised format where individual tokens are separated by spaces, no validation was done to check if each token is valid and meaningful (e.g. *?Such*, *:Yet*, *.O* were considered as one word/token each and included in all analysis).

THE END