Table of Contents

Introduction:

Stylometry is the analysis of differences in literary style between one writer and another. In this project we are going to study some of the writings of William Shakespeare and Christopher Marlowe.
"One of the classic authorship attribution studies was whether William Shakespeare wrote all his works, in particular one of his popular plays — Henry VI Trilogy — was highly disputed to have been written or co-authored by Christopher Marlowe."

Below is our data set for the simple stylometric analysis.

Writer	Title of the Text	File Name
Marlowe	Edward II	Edward_II_Marlowe.tok
Marlowe	The Jew of Malta	Jew_of_Malta_Marlowe.tok
Shakespeare	Richard II	Richard_II_Shakespeare.tok
Shakespeare	Hamlet	Hamlet_Shakespeare.tok
Shakespeare	Henry VI Part 1	Henry_VI_Part1_Shakespeare.tok
Shakespeare	Henry VI Part 2	Henry_VI_Part2_Shakespeare.tok

Instructions:

1. We now have 5 python files (preprocessor_29498724.py, character_29498724.py, word_29498724.py, visualiser_29498724.py & main_29498724.py) in this folder. Each of these files perform tasks according to the explanation below.
2. Please launch PyCharm software on your desktop.
3. Click on "OPEN".
4. Navigate to "29498724_A3" folder and click on open. You will be able to view multiple Task tabs once you click on the preprocessor_29498724.py/ character_29498724.py/ word_29498724.py/ visualiser_29498724.py and main_29498724.py files. This can help you to view the program with code comments before every code line, comments will help you understand the program logic.
5. You can Execute the program by right click on the respective task tab and click on RUN'****'. You will notice the console open to the bottom of the screen and you can now enter your input to obtain desired output.
6. Program instruction and output will be displayed on the same console.
7. If you would like to re-execute a program, you can do so by repeating step 6.

Task 1:

I start with creating a class ‘Preprocessor’ which contains methods to tokenise the text file. This task will convert the complete text in the file to a list of all the words to help our future analysis of characters and words.

Program:

- Initiated the reference tokenise list under the constructor for this class.
- Overload the print function to display each element of the list in readable format.
- Define a method called ‘tokenise’ which accepts the input string to produce a tokenised list.
- Splitting the input sequence over space.
- Return the tokenised list.

Task 2:

This Task emphasizes on counting the number of occurrences for each of the letters and numerals from the decoded sequences. We again define a class to filter the punctuation count and return it.

Program:

- Create a class 'CharacterAnalyser' which contains methods to count each character in tokenised list.
- In this task we use the dataframes/series from Pandas package for our analysis.
- Initiated the reference character dataframe under the constructor for this class
- Overload the print function to display dataframe in readable format.
- Define a method to analyse the count of each character in the list of characters and return the occurrence of each character as a dataframe.
- Return a dataframe of occurrences.

Task 3:

This Task performs counting the number of occurrences for each of the English words from the tokenised list.

Program:

- Create a class 'WordAnalyser' which contains methods to count occurrence each word in tokenised list.
- In this task we use the dataframes/series from Pandas package for our analysis.
- Initiated the reference character dataframe under the constructor to store word analysis.
- Overload the print function to display dataframe in readable format.
- Define a method to analyse the count of each word in the list of words and return the normalised occurrence of each character as a dataframe.
- We also import and filter the set of stopwords from a desired web-page to copare and count the occurrence of each stop word in the file and return the normalised occurrences.

- Introduce another method to count the word length used in each file and return the normalised word length in a dataframe.
- Return a dataframe of occurrences accordingly.

Task 4:

This Task deals with plotting the graph from all the analysis done in the last 3 tasks and help us visualise the data outcomes.

Program:

- Create a class 'AnalysisVisualiser' which contains methods to plot the analysis performed.
- Initiated the reference character list under the constructor to store word analysis.
- Define a method to take the instance variable and plot the graph (Character vs Normalised Count).
- Define a method to take the instance variable and plot the graph (Punctuations vs Normalised Count)
- Define a method to take the instance variable and plot the graph (Stopword vs Normalised Count)
- Define a method to take the instance variable and plot the graph (Word_length vs Normalised Count)

Task 5:

This task focuses on extracting input from a text file and to channelize the analysis process.

Program:

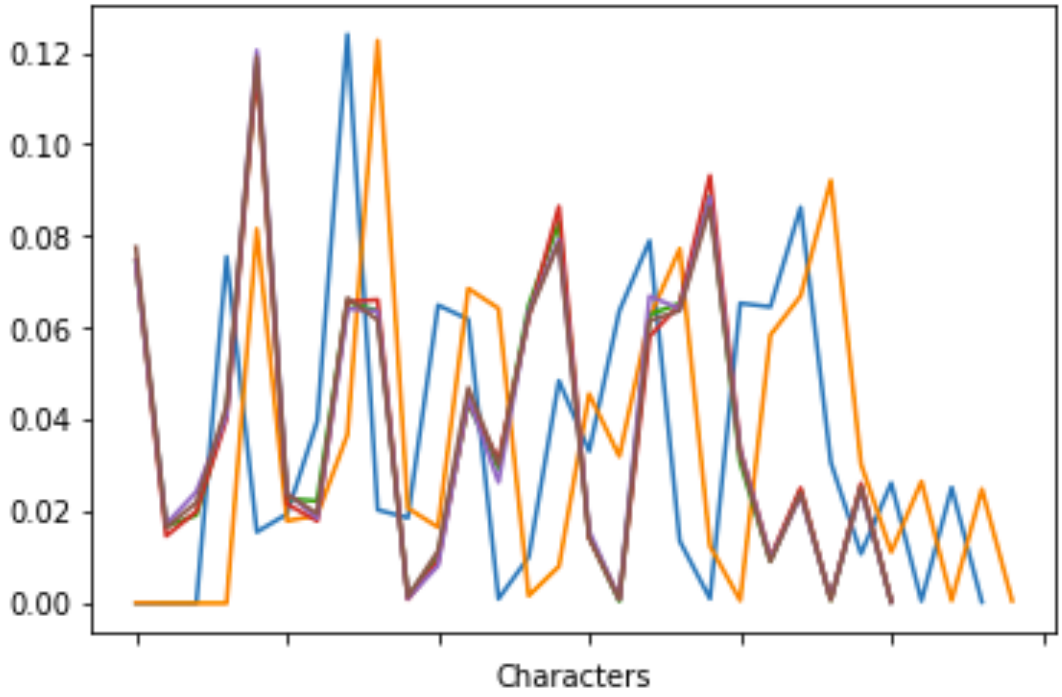
- Create a main function to accept input and to create instances to perform the class operation defined previous tasks.
- Create a main function
- Read all the text files to a variable for further analysis.
- Create a list of all the input variables to assist with iterations.
- Iterating through the input list to call methods and perform the core analysis to obtain analysis.
- Creating instances to capture the analysed data from the above methods.
- Creating an instances of all the analysis to plot a graph.

Assumptions:

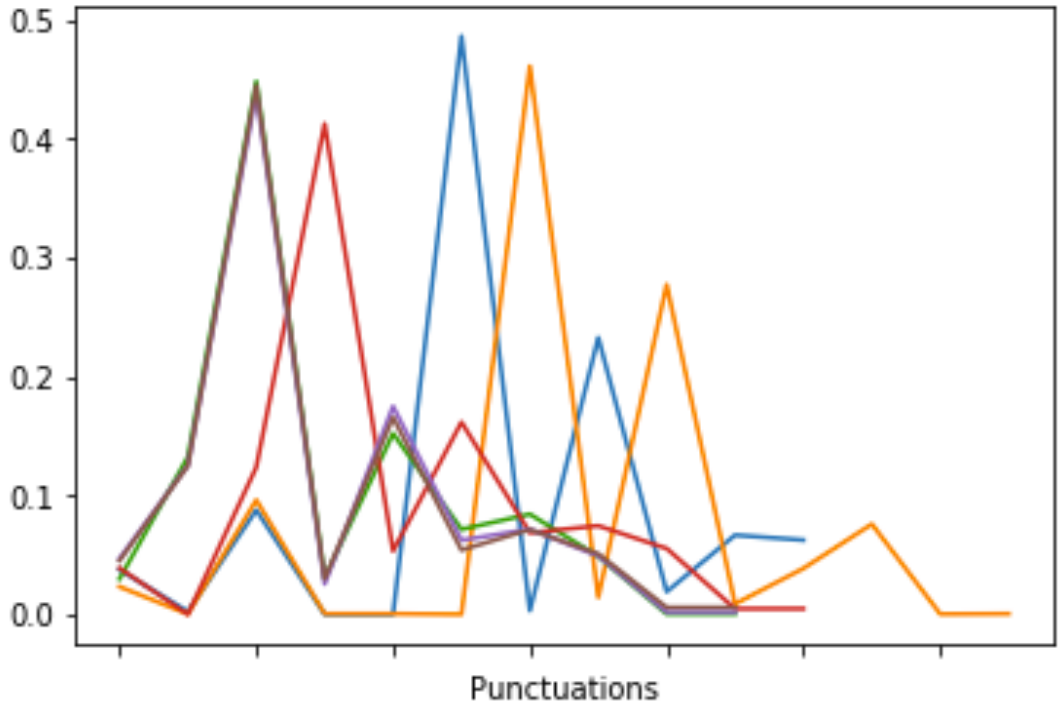
1. Expect the user to have pandas/matplotlib packages installed on his pycharm. Or basic knowledge to use Spyder.
2. Need to possess the access to internet to retrieve the stopwords list from web.

Expected output:

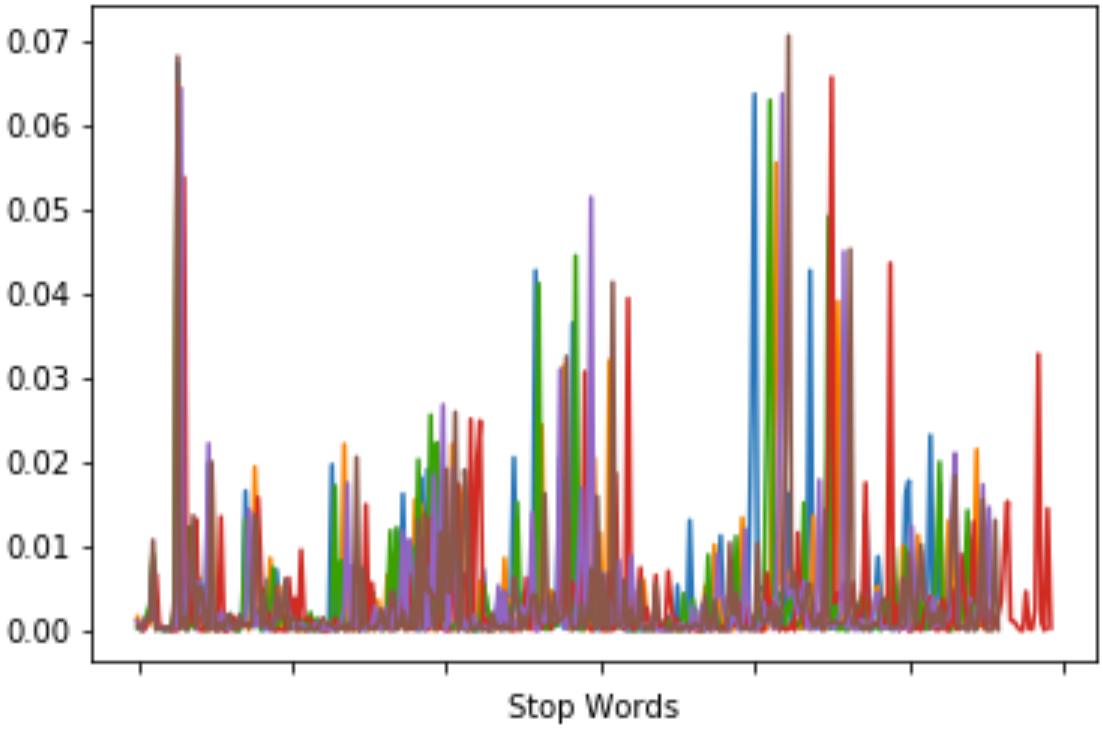
Character Analysis:



Punctuation Analysis:



Stop-word Analysis:



Word Length Analysis: