Name: Jemil Patel

Instructor: Gary Thai

CMSC 206 (CRN: 46518)

Project 4 Write-Up

***Project Design***

* Create a function named strip\_punctuations that accepts a list of words and returns an updated list after removing punctuations from before and after each word
  + Use two for loops to loop through each word and consecutively find the first and last occurrence of any letter in the word
  + Append the original word sliced from start to stop to the new list
  + Return the new list of words
* Create a function named average\_word\_length that accepts a list of words and returns the average length of a word
* Create a function named word\_frequency that accepts a list of words and returns a dictionary with (word, frequency) pair sorted in descending order by frequency
  + Using a for loop check if a word is not ‘a’, ‘an’, ‘the’, ‘and’
    - If the word is not already present in the dictionary, then assign that word as a key to the dictionary with a value of 1
    - Else increment the value of that word by 1
  + Create and return a new dictionary by sorting the original one via descending order of frequency
* Create a function named word\_length that accepts a list of words and returns a dictionary with (word, length) pair sorted in descending order by length
  + Using a for loop check if the word is not already present in the dictionary
    - If not, then assign that word as a key to the dictionary with a value equal to its length
  + Create and return a new dictionary by sorting the original one via descending order of length
* Create a function named number\_of\_sentences that accepts a text or a list of words and returns the number of sentences present
  + Using a for loop check if any word has ‘.’, ‘?’, ‘!’
    - If so, then increment the number of sentences by 1
  + Return the number of sentences
* Ask the user to input a valid filename and display an error message if file not found, repeat this until you get a valid filename
* Extract words from the text file
* Declare and initialize necessary variables to store different values using the above created functions
* Display your calculations
* Create a new variable named results to store the output from getmeasures method of readability package after passing the text of the speech as an argument
* Display the readability results
* Create a word cloud using the generate\_from\_frequencies method of the WordCloud class by passing the dictionary of word frequencies as an argument
* Display the word cloud and save it as a png file

***Test Cases***

Test Case 1

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Test Case 2

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

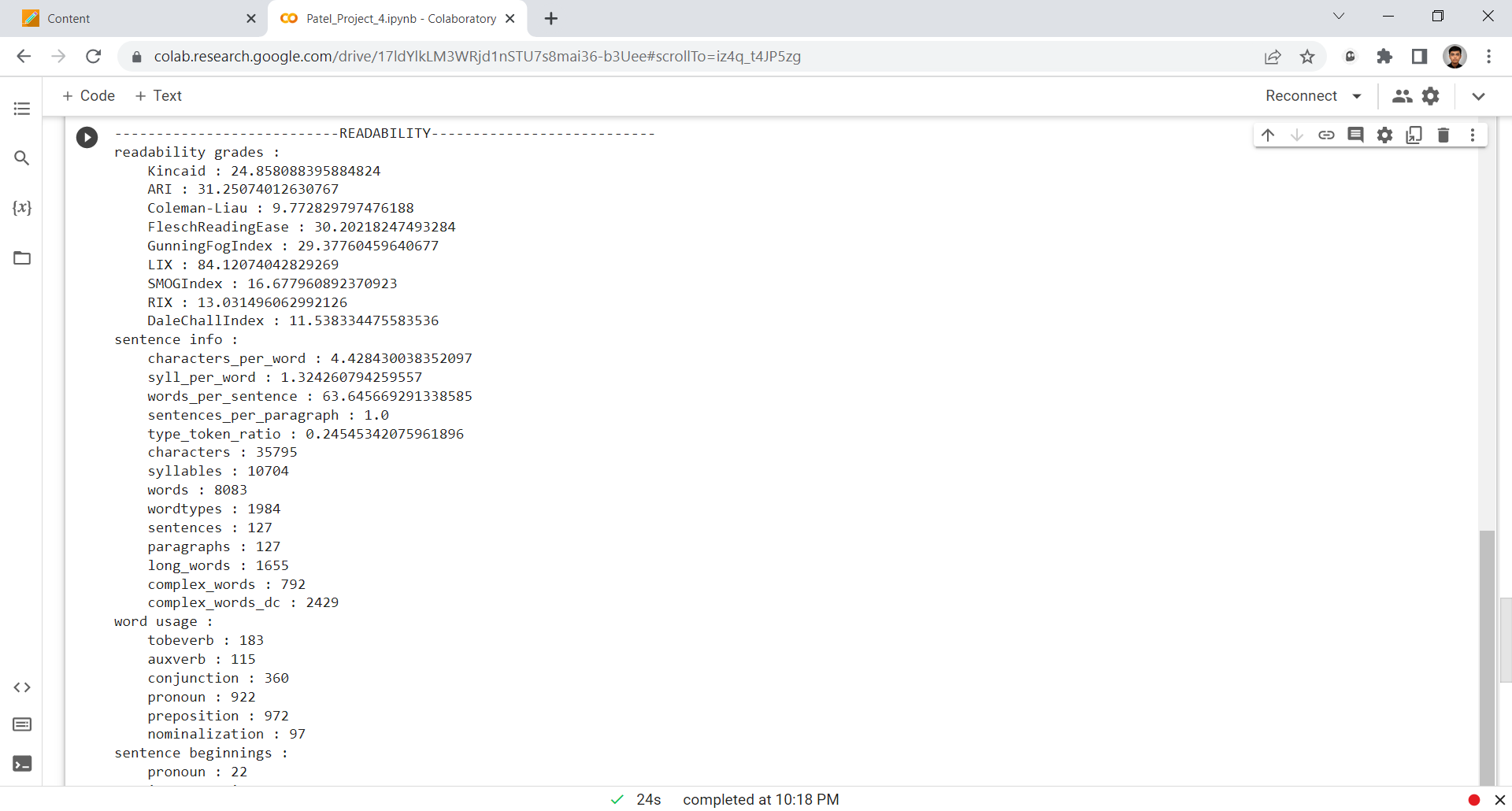
Test Case 3

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated



Graphical user interface, application, Word

Description automatically generated

Test Case 4

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

***Learning Experience***

Project 4 has been a great learning experience for me. Through it I learned a lot of new things such as readability package, wordcloud, etc. In order to start with it I first decided to break a speech into words separated by spaces and then strip away the punctuations before and after each word. Next, I created functions for each task wherein I basically just played and manipulated with a list of words to get a certain thing. This was it for the calculation part but for the readability score I had to read through the documentation of readability package to figure out how it works. I then implemented it in my program so that it provides a variety of details regarding a speech. Finally, I completed the wordcloud tutorial given to me which helped me create a wordcloud based on the frequency of words appearing in a given speech.

I struggled at many places, but it was due to my lack of logic, trial and error, and missing out something tiny that screws the whole program. I figured out on my own how to create each function, what data type will it utilize, and how will it work. I used the sorted function to sort a dictionary by descending order of its values. Unpacking the result from readability score was tedious but I managed to present the information in a neat and organized manner. Thus, I would say that this project was not impossible to complete but yes it certainly required some knowledge and effective thinking.

I will follow the same approach for my future projects that I used for this and my previous projects which is to read all the instructions carefully, form a rough base program, test it with many cases, and finally add the other requirements one by one. If I were the instructor, I would not change a thing because my program fulfils all the given requirements.

***Assumptions***

* The program is case-insensitive for any speech.
* Compound words will not be counted in the list of longest words.
* One cannot proceed without entering a valid filename.
* It was not possible for me to come with a specific expected output each time, so I just included the test run screenshots and did not create a table.