SCI1022 Python Assignment 3: Experimental evaluation of Zipf's law

In this assignment you have to develop code able to evaluate, experimentally, to what extent texts written in natural languages (e.g., English) fulfill the so-called Zipf's law. The law is named after the American linguist George Kingsley Zipf, who studied statistical occurrences in different languages.

Zipf's law statement

Zipf's law describes a relationship among **ranks** and **frequencies** of words in natural languages. Given a sample text (e.g., a book), and a word within the text, the **frequency** of the word is defined as the number of occurrences of the word within the text. The **rank** of a word is defined as the position of the word in a ranking of words by frequency (in descending order). Thus, the most common word, i.e., the one with highest frequency, has rank 1, the second most common word has rank 2, and so on.

If we denote by r the rank of a word, and by f its frequency, then, accordingly to Zipf's law, we have that:

$$f(r) = cr^{-s},$$

where c and s are parameters which depend on the particular text and language. It turns out that, for most texts and languages, s is (almost) 1, i.e., f is (almost) inversely proportional to r. This means that the frequency of word with rank 2 will be approximately half the one of the most common word, the one with rank 4 approximately a factor of 4 smaller, and so on. **In words, Zipf's law states that only a few words are used very often, while many or most are used rarely.**

Although Zipf's law was first observed empirically in the field of quantitative linguistics, the same relationship occurs in many other rankings of human created systems, such as the ranks of mathematical expressions or ranks of notes in music, and even in uncontrolled environments, such as the population ranks of cities in various countries, corporation sizes, income rankings, among others.

Plotting rank-frequency curves

In order to evaluate in a plot whether a measured rank-frequency relationship actually follows Zipf's law, it helps to take the base 10 logarithm on both sides of the equation above, leading to:

$$\log_{10}(f(r)) = \log_{10}(cr^{-s}) = \log_{10}(c) - s\log_{10}(r).$$

Thus, if we rename $y = \log_{10}(f(r))$ and $x = \log_{10}(r)$, and plot y versus x, then we should get a straight line with slope $-s$ and intercept $\log_{10}(c)$. (Recall that the general equation of a straight-line $y = g(x)$ is $y = n + mx$, where n is the intercept and m is the slope.)

Texts subject to analysis. File format explained

We will analyze four different classic books written in English, namely:

- From the Earth to the Moon, by Jules Verne
- Time Machine, by Herbert George Wells
- The Picture of Dorian Gray, by Oscar Wilde
- The Adventures of Sherlock Holmes, by Arthur Conan Doyle

The source of these books will be Project Gutenberg, an electronic library with more than 60,000 Ebooks (at the date of writing). Project Gutenberg's library only includes "public domain" works that are out of copyright. You will find the pure text file (ASCII) corresponding to each of the four books in a folder available at the Moodle page of the unit. **You have to download these files in the same folder of your system where you downloaded this notebook.**

As usual, before writing code to read a text file, you must know the format of the file. The file format of Project Gutenberg's plain text Ebooks is actually very simple. There are three sections in sequence: header, body, and tail. **The body contains the actual text of the book.** Thus, we must skip the contents of the header and the tail in our analysis. To this end, the header and body are separated by a line which starts with the string `"*** START OF THIS"`, while the body and the tail, are separated by a line which starts with the string `"*** END OF THIS"`. At this point, you are highly encouraged to open, in a text editor (e.g., Notepad in Windows, or TextEdit in MacOSx), the books that you downloaded, in order to double check that this indeed the format.

A frequency table, main program data structure

The main data structure of the code that you have to write is a frequency table. A frequency table is built out of a given text. It is a data structure which, given a word, provides its frequency in the text. The first big decision to be made in this assignment is: which is the most appropriate Python data structure to store the frequency table? An appropriate data structure should let you efficiently add new elements to the table in a dynamic way (this excludes NumPy arrays), and also should be able to provide the frequency associated to a word in time independent on the number of elements of the table (this excludes lists).

> **Task 1 (2/22 points).** Write a function that given an already opened file object, modifies the file object such that the file object is positioned in the first line of the body of the book. The function is though not to be a fruitful function, i.e., it does not return a value. **Hint**: you may consider the `startswith` method of type `str` in order to solve this task. (Recall that you can ask for help calling `help(str.startswith)` on a code cell.)
>
> **Task 2 (4/22 points).** Write a function that given a line in the body of the Ebook (as a string), and the frequency table, processes the line and updates the frequency table conformally with the contents of the line. The function is though not to be a fruitful function, i.e., it does not return a value. Some hints and considerations:
>
> - Before splitting the line in words, replace hyphens (i.e., `"-"`) by blank spaces (i.e., `" "`). You may consider useful to use the `replace` method of type `str`.
> - After splitting the line in words, the resulting words may have leading and/or trailing punctuation signs or blank spaces. The `string` module provides the `punctuation` variable, which contains all English punctuation signs. Figure out how you can use the `strip` method of `str` and such a variable in order to get rid of the leading and/or trailing punctuation signs in each word.
> - We will neglect the case of letters in our analysis. Thus, for example, we will consider "The" and "the" to be the same word. Thus, before accessing the table, you must transform all letters of the word into lower case. Figure out how the `lower` method of `str` can be helpful for such purpose.
>
> **Task 3 (4/22 points).** Write a function that given a file's name of a plain text Gutenberg's Ebook, returns its associated frequency table. The function MUST use the functions written in Task 1 & 2.
>
> **Task 4 (4/22 points).** Write a function that given a frequency table, returns a list of `(frequency,word)` pairs (i.e., `tuples` of two elements) in descending order by `frequency`. Write in a text cell the answer to the following questions: Which are the top-3 words and associated frequencies in the four books subject of study? Describe in your own words how does the frequency decay with rank for the words in the top-3. **Hint: you may consider the `sort` method of type `list` in order to solve this task. (Recall that you can ask for help calling `help(list.sort)` on a code cell.)**
>
> **Task 5 (2/22 points).** Write a function that given the list generated in Task 4, returns two lists with `log10(rank)` and `log10(frequency)`, respectively, for all words in the text.
>
> **Task 6 (2/22 points).** Write a function that given a list of frequency tables, and a list of labels (strings) with as many labels as tables, generates a plot of `log10(rank)` versus `log10(frequency)` for each of the frequency tables in the list. The curves in the plot must be labeled as given in the second argument. Note that thanks to the `log10`-trick, we do not need a log-log scale plot, i.e., linear scale suffices for both axes. Use meaningful labels for the x-, y-axis and the title of the plot. **Hint:** You may consider useful to adapt the plot statements given in the Jupyter notebook of Assignment 2 to solve this task.
>
> **Task 7 (2/22 points).** In order to help you out in determining to which extent the curves obtained in Task 6 fulfill Zipf's law, modify the function in Task 6 such that it additionally draws several straight lines with different slopes (i.e., values of s), e.g., `-1`, `-1.2`, `-1.5`, `-2`, and intercept equal to `2`.
>
> **Task 8 (2/22 points).** Discuss with your own words in a text cell the plot generated using the function obtained in Task 7. Which is the value of `s` among the ones evaluated that best fits the measurements for each of the books. Overall, would you say that the general trend observed for the books subject of study satisfy Zipf's law? Include any observation that you consider relevant to your answer.

```
In [ ]: # Solution to Task 1 goes here
```

```
In [ ]: # Solution to Task 2 goes here
```

```
In [ ]: # Solution to Task 3 goes here
```

```
In [ ]: # Solution to Task 4 goes here
```

Write your answers to the questions in Task 4 in this cell.

```
In [ ]: # Solution to Task 5 goes here
```

```
In [ ]: # Solution to Task 6 goes here
```

```
In [ ]: # Solution to Task 7 goes here
```

Write your answers to the questions in Task 8 in this cell.