DATA.STAT.840 Statistical Methods for Text Data Analysis
Exercises for Lecture 2: Basic text processing

Exercise 2.1: Data acquisition - Building a better web crawler.
The web crawler code shown on Lecture 2 is poor in at least two respects:

1. It can crawl the same page multiple times, if a link on a later crawled page points to the already-crawled page.
2. It inserts all links from each page in order as pages to be crawled. If some page contains thousands of links, the crawling will crawl those first and may never get to the links from the next page, especially if the total number of pages are limited.

Suggest how to create an improved web crawler that does not have these problems. Provide in your answer the improved code, either as pseudocode or as real code in Python or in another programming language. The code must be detailed enough to show how the problem is solved using the data types and structures of the language you have chosen.

Exercise 2.2: Data acquisition and processing from Project Gutenberg.
In this exercise we will process top-downloaded public domain ebooks from Project Gutenberg.

(a) Create a variant of the web crawler which is intended to download the top-k most downloaded ebooks of the last 30 days from Project Gutenberg in .TXT format.
(b) Using the crawler, download the top-20 ebooks (k=20). Report the names and addresses of the books.
(c) Use the processing pipeline described on the lecture to tokenize and lemmatize the downloaded books.
(d) Create a unified vocabulary from the ebooks; report the top-100 words.

Hint 1: the top-100 downloaded books of the last 30 days Project Gutenberg are listed on the site. In your answer, report the required statistics as well as your program code. You can do this exercise either in Python or in any other language of your choice.

Hint 2: Sometimes domain-specific processing is needed. Here, project Gutenberg ebooks include a header-text in the beginning and a footer-text (license description) at the end. For best results, you can modify the processing code to add a domain-specific processing step that removes the header and footer texts from each ebook. This is optional for this exercise, but should give better results. How to do it:

- The header-text ends with a line "*** *START OF THIS PROJECT GUTENBERG EBOOK ???????? ****", and the footer-text (license description) starts with a line "*** *END OF THIS PROJECT GUTENBERG EBOOK ???????? ****", where ???????? is the name of the ebook. The actual book content is between these lines.
- Once you have read in the text string of an ebook, you can find the location of "*** *START OF THIS PROJECT GUTENBERG EBOOK*", and then find the next newline - the book content starts after that. Similarly, you can find the location of "*** *END OF THIS PROJECT GUTENBERG EBOOK*" - the book content ends before that.
- Use Python's find function mytextstring.find(mysubstring,startindex,endindex) where mytextstring is the string where you want to search, mysubstring is the string you want to find, startindex is a zero-based index of where to start looking, and endindex is the final index where you want to look. The function returns the index where the substring was found (or -1 if it was not found).
- Once you have found the start and end indices of the content of the ebook, you can just take that substring as your book content: mybookcontent = mytextstring[startindex:endindex]

Exercise 2.3: Zipf's law.

Use the top-20 books from Project Gutenberg to examine whether Zipf's law holds here too.

(a) Compute a plot of the frequencies of all words in the vocabulary (= count of each word, divided by the total count of all words together), sorted by frequency. Report the plot.

(b) Then plot the frequency distribution according to Zipf's law (slide 36 of lecture 2) in the same plot as the real frequencies. Try several choices of the exponent a, and try to find one that in your subjective opinion fits best. Report the plots for different choices of a and your choice for best a.

In addition to the plots, report also your program code. You can do this exercise either in Python or in any other language of your choice.