The University of Melbourne
BUSA90543 TEXT ANALYTICS FOR BUSINESS

Practical exercises: Week 1

1. Follow this instruction on using Jupyter notebook and Python on LMS.
2. Solve the Matching Email and Matching Phone Number problems on *RegexOne*.
3. On the LMS, there are two HTML files — `Gf-wiki.html` and `Gf-imdb.html` — taken directly from the Web.[1] Upload them to the Colab environment. Let's first think about pre-processing the HTML formatting in the first part of the iPython notebook `TWA-week1.ipynb`.
   (a) What pages do the HTML files correspond to? Try to read the files as if they were plain text. (Warning: Spoilers!)
   (b) What do the given Python codes do? Why would we wish to do this?
   (c) Choose one of the codes and run it. The output should be written to `wiki-out.txt` — but it is (arguably) doesn't accomplish our goal of stripping the HTML formatting. Why not?
   (d) Alter the code so that the output is only the visible textual elements. (There is a hint for how to do this in the lecture slides!)
4. The next code uses the NLTK method `word_tokenize()` based on the output of the previous code. (If you haven't downloaded NLTK's data, you might skip this step.)
   (a) How many times does the word "Brando" appear in the two files?
   (b) What is the most common word in each file? Why is this the case?
5. A simple tokenisation strategy for English documents was given in the lectures, as follows:
   - Strip formatting
   - Strip punctuation
   - Fold case
   - Break at whitespace
   - Stem
   - Remove stop words

   (a) Alter the code for the previous question so that it uses this strategy. (There are some commented lines which might help you.)
   (b) Compare your program to the tokens produced by `word_tokenize()` in NLTK.
      i. Does the count of "Brando" change? Why?
      ii. Does the length of the word list change? Why?
      iii. Which word is the most frequent now?
   (c) A better solution to the problem of finding the most frequent token(s) uses a dictionary:

```
words = {}
for token in tokens:
    if token not in words:
        words[token] = 1
    else:
        words[token] += 1
```

   Alter the code so that it makes use of a dictionary to count the word frequencies. More on this next week!

[1] We've removed some scripting to improve readability and aid in processing. We're also claiming fair dealings to use these files in an educative capacity, under the Australian Copyright Act (1968, as amended in 2018).