Home Work #2

1. A common method of text analysis is to represent a document as a *bag of words* and characterize the frequency of occurrence of individual words in isolation. However, that neglects the insight we get from the relationships between words. In particular, which words tend to occur right next to others are worth examining.

   In this assignment, you will create a Python script **(without using any libraries)**[1] in a file named `<yourfirstname>.py` executable using Python 3, that reads in an input text document file and creates a sorted list of bigrams (2-grams). Your script should

   (a) use a boilerplate invocation, as before, directing control to a function `main()`, which should extract the filename from the command line;

   (b) consider all words in lower-case;

   (c) assume all words English words terminated by spaces or standard punctuation (comma, colon, and semicolon), hyphenated words are single words; and each sentences is terminated with a period;

   (d) ignore *stop-words* (listed in the given file `stop-words.txt`) and numerals;

   (e) count $f(w_i, w_j)$, the number of occurrences of word w_j immediately following w_i in the text but without crossing sentence boundaries;

   (f) compute the sentiment score $s(w_i, w_j)$ for each pair of words with non-zero f-values: s is defined as

   $$s(w_i, w_j) = \begin{cases} -m(w_j) & \text{if } w_i = \text{'no' or } w_i = \text{'not'} \\ m(w_i) + m(w_j) & \text{otherwise} \end{cases}$$

   where $m(w)$ is the sentiment value of the word w as per the data file `part-AFINN-only-words.txt` (0 if w is not listed);

   (g) sort the pairs of words using f and s as the primary and secondary sorting keys respectively;

   (h) print the top K entries (or all if you have fewer than K) of your sorted word pairs along with their f and s values in neatly aligned columns. Take $K = 20$; and

   (i) output into a file `<yourfirstname-out.csv>` all the entries of your sorted word pairs along with their f and s values, one word-pair, e.g.,

   ```
   word1, word2, frequency, score
   absolutely, beautiful, 2, 3
   not, worried, 1, 3
   ```

   Consider using Python string functions `split()`, `rstrip()`, and dictionaries.

[1] There is one exception: `sys`, because you will extract the input document file name from `sys.arv[1]`.