Assignment 05:
Processing Frequency Lists

handed out: January 12, 20:00
to be submitted by: January 19, 20:00

In this assignment, you are going to practice file input and output, with a token frequency list as an example. A very useful collection of frequency lists was extracted by Hermit Dave from the OpenSubtitles corpus (`http://opus.nlpl.eu/OpenSubtitles.php`), a corpus of parallel subtitles for thousands of movies which, due to the high amount of dialogue in movies, comes a lot closer to actual every-day language usage than e.g. frequency lists extracted from the Wikipedia or newspaper texts. The frequency lists for dozens of languages are available in this public repository, which is also where the example frequency list for this exercise was taken from:

`https://github.com/hermitdave/FrequencyWords/`

Task 1: Reading the Frequency List from a File [3 points]

The tokens in the frequency file are ranked by their absolute frequency in the corpus, and each token is given on a separate line, with the corpus frequency (an integer) following after a space. Your first task is to write a method `read_frequency_file(freq_file_name)` which reads in a frequency file with name `frequency_file_name` and stores the frequency information in a simple data structure. Every token is to be stored in a tuple $(token, count)$, and the result of `read_frequency_file(freq_file_name)` should be a list of such tuples in the order in which they appear in the file. After a successful implementation, you should be able to extract the three most common words of English (and their counts) like this:

```
>>> frequencies = read_frequency_file("freq_en_50k.txt")
>>> frequencies[0:3]
[('you', 22484400), ('i', 19975318), ('the', 17594291)]
```

Task 2: Determining the Decile Thresholds [3 points]

Next, we need a function `determine_decile_thresholds(frequencies)` that splits the language's tokens into ten parts of equal frequency. The **deciles** or 10%-level thresholds are the number of word forms that are needed to cover 10% of a text, 20%, 30%, and so on to 90%. The correct result on the English data should imply that 72 word forms are enough to know 50% of all tokens in an English text, whereas to reach 90% of all tokens (a threshold frequently quoted as the threshold to pleasant reading in the literature), passive knowledge of 4,533 English word forms is necessary:

```
>>> thresholds = determine_decile_thresholds(frequencies)
>>> thresholds[4:9]
[72, 142, 330, 968, 4533]
```

Task 3: Computing Prefix Frequencies [3 points]

Another interesting thing you can do with a frequency list is to compute the frequencies of each prefix. A prefix of a string is each substring that starts at the start of the string, including the string itself. For example, the prefixes of `"house"` are `"h"`, `"ho"`, `"hou"`, `"hous"`, and `"house"`. Write a function `get_prefix_frequencies(frequencies)` which takes a list of (*token*, *count*) pairs as produced by Task 1, and creates a similar list containing all the prefix counts. If you did this correctly, you should be able to do this on your interactive console:

```
>>> prefix_frequencies = get_prefix_frequencies(frequencies)
>>> prefix_frequencies.sort()
>>> prefix_frequencies[10383:10387]
[('bist', 691), ('bistr', 576), ('bistro', 576), ('bit', 337003)]
```

Task 4: Save Prefix Frequencies in File [3 points]

Your final task is to write a function `store_frequencies_alphabetically(frequencies, freq_file_name)` which writes the frequencies from any list of string-frequency pairs into a new alphabetically sorted file with the name `freq_file_name`), in the same format as the file you read the frequencies from. In order to make the output cleaner, only the prefixes which occurred 1000 times or more are supposed to be written to the file. Use the function to save the result of your prefix frequency computations into a new file with name `freq_en_prefixes.txt`. If you did everything correctly, the first few lines of your result file should look like this:

```
a 45546869
aa 55975
aaa 7478
aaaa 2175
aaaah 1092
aaah 3973
aah 35842
aar 9254
aaro 7657
aaron 7657
```

That's it! Before submitting, don't forget to test your methods one last time using `test_ex_05.py`, and checking whether the first lines of the output are identical to the desired result! This time, you must submit TWO files: your version of `ex_05.py`, and the output file `freq_en_prefixes.txt`.

Total points: 12