Exercise sheet 4

February 18, 2020

Exercise 1 *(12 points)*
The aim of this exercise is to compare the effective runtime of two different methods to count the number of word occurrences. The first method will use a sorted list of words for counting their occurences while the second method will use a hashmap for the counting part. To accomplish this you will use a given function to read in words from a text file and then compute the most common word as well as the 5 most common words in this file for both methods. Use the template on our website (*compare_ count_ methods.py*) with the predefined function *read_ word_ list()* to read in the data file *data.encrypted.txt*. The read-in words are returned as a list and are ready to be processed by you. Use the remaining functions in the template and implement the needed functionality.
ATTENTION:
DO NOT UPLOAD THE *data.encrypted.txt* FILE INTO THE SVN REPOSITORY!
Hints:

- As mentioned on Exercise Sheet 2, Python uses call-by-reference when passing objects to functions. Since we want each function to work on the same input list when comparing runtimes, you therefore need to make a copy of the list before passing it to a function. In our case we want to sort the list, for which you can use the built-in function *yourlist.sort()*. After sorting the list, each distinct word will be grouped together. Use this to your advantage and count the size of each group to determine the occurrence count of each word. Compute the most common as well as the 5 most common words (see given functions for more details) and print them to the console.

- When counting words with a hashmap (we use Python's dictionary data type for this), insert a new entry with a count of 1 into the map for each new word. For each repeating occurence just increment the counter. After counting, you can convert the dictionary into a list of tuples (word, count) using *list(yourmap.items())*. For writing tests, you can use our *test.encrypted.txt* file.

Exercise 2 *(8 points)*

1. Calculate the runtimes of the four functions from the first exercise by implementing the *measure_runtimes_top_count()* and *measure_runtimes_top_x_count()* functions in the template. Inside run each function 5 times and report the average runtime for each of the four functions. Make sure not to use a previously sorted list when repeatedly running the functions.

2. Discuss your results regarding the runtime difference. Compare the two approaches (sorted list vs. hashing) as well as the top_count vs. top_x_count versions of both approaches. What are your expectations?
   Which algorithm is faster and why do you think this is the case?

Commit
Commit your code into the SVN in a new subdirectory **uebungsblatt_04**, together with a *discussion.txt* file for exercise 2. Commit your feedback in a text file *erfahrungen.txt* as usual. There specify which tasks have been difficult for you and where did you have problems? How much time did you spend to solve the exercises?