CS 8001 Big Data

HW #1: Word Count (10 points + 2 bonus points)

Jing Su (Graduate)

12519623

1. (10 points) the 10 most common words in the first 3 sections of "The Sonnets" (about 330 words in total) from the Complete Works of William Shakespeare retrieved from Project Gutenberg and the number of occurrence of each of them.

1) A brief description of your Python environment, code structure, and the execution process of the code.

Python Environment: Spyder(Python 3.5)

Code Structure: I use a "for" loop to to split each word from the text file and count their appearance in the loop. Afterwards, I use two nested loops to find top 10 words that their values match the top 10 values.

Execution Process: The code has two parts. The first part I split each words from the file and count the appearance of each word. I use dictionary, wordcount{}, to store the key-value pairs. Then, the second part I use heapq to get the top 10 values and print those key-value pairs.

2) Output of your code (10 most common words and their counts).

```
In [7]: runfile('E:/MU课件/big data algorithm/hw1/top10wordcount.py',
wdir='E:/MU课件/big data algorithm/hw1')
thy 14
the 13
thou 8
of 7
and 6
to 6
thine 5
in 4
by 4
And 4
Execution time: 0.0026250049266083827
```

3) Execution time of your code in seconds.

```
Execution time: 0.0026250049266083827
```

4) Your code with appropriate comments.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 22 00:06:06 2016
@author: Jing Su
"""
import heapq
import time

start = time.clock()              # start time of CPU
#open the frirst 3 sections
file=open("E:/MU课件/big data algorithm/hw1/First3Section.txt","r+")
wordcount={}
for word in file.read().split(): #split each word from the article
    if word not in wordcount:
        wordcount[word] = 1      #set wordcont=1 if the word appear once
    else:
        wordcount[word] += 1     #set wordcont+=1 if the word appear again
#for key in wordcount.keys():
        #print the key-value pairs of all words
        #print ("%s %s " %(key , wordcount[key]))

topNum = 10
#get top 10 words that have largest values
nlargestList = heapq.nlargest(topNum, wordcount.values())
for value in nlargestList: #find the top 10 values
    for key in wordcount.keys(): #find keys that have largest values
        if wordcount[key] == value:
            print ("%s %s " %(key , wordcount[key]))
            #Set wordcount[key]=0 after the word printed
            #So the word won't be printed the second time
            wordcount[key] = 0

file.close();  #close the file

end = time.clock()  #end time of CPU
print ("Execution time:",end-start) #print CPU execution time
```

2. (2 bonus points) the 20 most common words in the Complete Works of William Shakespeare retrieved from Project Gutenberg and the number of occurrence of each of them.

1) A brief description of your Python environment, code structure, and the execution process of the code.

Python Environment: Spyder(Python 3.5)

Code Structure:I use a "for" loop to to split each word from the text file and count their appearance in the loop. Afterwards, I use two nested loops to find top 20 words that their values match the top 20 values.

Execution Process:The code has two parts. The first part I split each words from the file and count the appearance of each word. I use dictionary, wordcount{}, to store the key-value pairs. The second part I use heapq to get the top 20 values and print those key-value pairs.

2) Output of your code (20 most common words and their counts).

```
In [8]: runfile('E:/MU课件/big data algorithm/hw1/top20wordcount.py',
wdir='E:/MU课件/big data algorithm/hw1')
the 23000
I 19392
and 18114
to 15441
of 15365
a 12424
my 10747
in 9463
you 9012
is 7788
that 7428
And 7021
not 6876
with 6675
his 6135
your 5938
be 5935
for 5571
have 5173
it 4858
Execution time: 1.391666176255427
```

3) Execution time of your code in seconds.

```
Execution time: 1.391666176255427
```

4) Your code with appropriate comments.

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 22 00:06:06 2016
@author: Jing Su
"""
import heapq
import time

start = time.clock() # start time of CPU
#open the Sonnets.txt
file=open("E:/MU课件/big data algorithm/hw1/Sonnets.txt","r+")
wordcount={}
for word in file.read().split(): #split each word from the article
    if word not in wordcount:
        wordcount[word] = 1  #set wordcont=1 if the word appear once
    else:
        wordcount[word] += 1 #set wordcont+=1 if the word appear again
#for key in wordcount.keys():
  #print the key-value pairs of all words
  #print ("%s %s " %(key , wordcount[key]))

topNum = 20
#get top 20 words that have largest value
nlargestList = heapq.nlargest(topNum, wordcount.values())
for value in nlargestList:  #find the top 20 values
    for key in wordcount.keys():  #find keys that have largest values
        if wordcount[key] == value:
            print ("%s %s " %(key , wordcount[key]))
            #Set wordcount[key]=0 after the word printed
            #So the word won't be printed again
            wordcount[key] = 0

file.close(); #close the file

end = time.clock() #end time of CPU
print ("Execution time:",end-start) #print CPU execution time
```