Lab2

August 7, 2020

[1]:
```
rdd = sc.textFile("/data/students/bigdata_internet/lab2/word_frequency.tsv")
outputPath = "lab2/"
```

[2]:
```
# How can you take a look at some of the lines of the file?
```

[3]:
```
somelines = rdd.take(5)
print('some of the line of the input rdd: ',somelines)
```

```
some of the line of the input rdd:  ['have\t338996', 'bought\t46988',
'several\t19688', 'of\t792000', 'Vitality\t252']
```

[4]:
```
# How many words does it contain?
```

[5]:
```
numlines = rdd.count()
print('Total number of lines (aka words) in the input rdd', numlines)
```

```
Total number of lines (aka words) in the input rdd 339819
```

[6]:
```
# Keep only the lines containing words that start with a prefix (a string) that␣
 ↪is a parameter of the script,
# i.e., a global variable PREFIX .
PREFIX = 'ho'
```

[7]:
```
# function to find the prefix and keep the line associated
def findprefix(line, Prefix):
    PREFIX = Prefix
    # quality is good    87367
    # the 'quality' prefix should be the first 0:len(PREFIX) values
    # i also don't care what comes after the orefix, it just has to be there
    if line[0:len(PREFIX)] == PREFIX:
        return True
    else :
        return False
```

[8]:
```
# creating the rdd that contains the lines filtered by the desired prefix
filtered_by_prefix_rdd = rdd.filter(lambda line: findprefix(line,PREFIX))
```

[9]:
```
print('Some lines of the filtered prefix rdd: ',filtered_by_prefix_rdd.take(5))
```

```
Some lines of the filtered prefix rdd:  ['hot\t32944', 'home\t17995',
'however\t12492', 'holds\t1762', 'hot"\t108']
```

[10]:
```
print('Number of lines in the filtered prefix rdd: ',filtered_by_prefix_rdd.
↪count())
```

```
Number of lines in the filtered prefix rdd:  1519
```

[11]:
```
# The result is the set of lines ( word\tfreq ) that satisfy the filtering␣
↪operation.
```

[12]:
```
# Print on the standard output:
    # The number of selected lines
    # The maximum frequency ( maxfreq ) among the ones of the selected lines (i.
↪e., the maximum value of
    # freq in the lines obtained by applying the filter).
```

[13]:
```
# function to split the text from the frequency values for each line
def splitline(line):
    description, freq = line.split("\t")
    return int(freq)
```

[14]:
```
# rdd containing only the frequency values
frequencies_rdd = filtered_by_prefix_rdd.map(splitline)
```

[15]:
```
print(frequencies_rdd.take(10))
```

```
[32944, 17995, 12492, 1762, 108, 1, 8668, 4730, 152, 6]
```

[16]:
```
# function to find the maximum value for all the frequencies
def findMax(f1,f2):
    if f1>f2:
        return f1
    elif f2>f1:
        return f2
    else:
        return f1
```

[17]:
```
# defining the maximm value of frequency
maxfreq_o = frequencies_rdd.reduce(findMax)
```

[18]:
```
print('The max frequency value is: ',maxfreq_o)
```

```
The max frequency value is:  36264
```

[19]:
```
# 1.2 Filter most frequent words
# Keep only the lines with a frequency freq greater than 0.8*maxfreq .
```

[20]:
```
# function to select the line that satisfy the condition on frequency
def keepsome(line, Maxfreq):
    maxfreq = Maxfreq
    description, freq = line.split("\t")
    freq = int(freq)
    if freq >= 0.8*maxfreq:
        return True
    else:
        return False
```

[21]:
```
# defining the rdd that contains only values which frequency satisfies the␣
 ↪threshold value
greater_than_rdd = filtered_by_prefix_rdd.filter(lambda line:␣
 ↪keepsome(line,maxfreq_o))
```

[22]:
```
print('Some values in the filtered by threshold frequency rdd:␣
 ↪',greater_than_rdd.take(10))
```

```
Some values in the filtered by threshold frequency rdd:  ['hot\t32944',
'how\t36264']
```

[23]:
```
# 1.3
# Count the number of selected lines and print this number on the standard␣
 ↪output
```

[24]:
```
print('Number of lines in the threshold rdd: ',greater_than_rdd.count())
```

```
Number of lines in the threshold rdd:  2
```

[25]:
```
# Save the selected words (without frequency) in an output folder for␣
 ↪inspecting the results
# the operation is
#greater_than_rdd.saveAsTextFile(outputPath)
```

[26]:
```
# 2.1
# Run you application locally on Jupyter web interface by using the PySpark␣
 ↪(Local) kernel
# and select only the lines containing the words starting with "ho" from the␣
 ↪the HDFS file
# /data/students/bigdata_internet/lab2/word_frequency.tsv .
# going to top and changing the prefix and kernel
```

[27]:
```
# Create a Python script to be executed with the spark-submit command. Note:␣
 ↪the prefix and
# the output_folder must be specified by two command line arguments (Hint: use␣
 ↪sys.argv[] ) .
```

```
# Run your script and select only the lines containing the words starting with␣
↪"ho"
# from the content of the HDFS file /data/students/bigdata_internet/lab2/
↪word_frequency.tsv .

# s274990@jupyter-s274990:~/newLabs/lab2$ spark-submit --master local␣
↪--deploy-mode client lab2_2.py ho lab2/
# WARNING: User-defined SPARK_HOME (/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.
↪p0.1425774/lib/spark) overrides detected (/opt/cloudera/parcels/CDH/lib/
↪spark).
# WARNING: Running spark-class from user-defined location.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4040. Attempting port 4041.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4041. Attempting port 4042.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4042. Attempting port 4043.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4043. Attempting port 4044.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4044. Attempting port 4045.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4045. Attempting port 4046.
# 20/08/07 10:19:10 WARN util.Utils: Service 'SparkUI' could not bind on port␣
↪4046. Attempting port 4047.
# opath:  lab2/
# prefix is:  ho
# some lines of original rdd:  ['have\t338996', 'bought\t46988',␣
↪'several\t19688', 'of\t792000', 'Vitality\t252']                              ␣
↪
# total number of lines in rdd:  339819
# some lines of the filtered rdd:  ['hot\t32944', 'home\t17995',␣
↪'however\t12492', 'holds\t1762', 'hot"\t108']
# number of lines in filtered rdd:  1519
# filtered_by_prefix_rdd:  ['hot\t32944', 'home\t17995', 'however\t12492',␣
↪'holds\t1762', 'hot"\t108']
# frequencies_rdd:  1519
# max freq:  36264
# number of elements found greater than 0.8*maxfreq:  2
```

[28]:
```
# on the cluster by using the --master yarn option of spark submit


# s274990@jupyter-s274990:~/newLabs/lab2$ spark-submit --master yarn␣
↪--deploy-mode client lab2_2.py ho lab2/
```

```
# WARNING: User-defined SPARK_HOME (/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.
 ↪p0.1425774/lib/spark) overrides detected (/opt/cloudera/parcels/CDH/lib/
 ↪spark).
# WARNING: Running spark-class from user-defined location.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4040. Attempting port 4041.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4041. Attempting port 4042.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4042. Attempting port 4043.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4043. Attempting port 4044.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4044. Attempting port 4045.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4045. Attempting port 4046.
# 20/08/07 10:21:06 WARN util.Utils: Service 'SparkUI' could not bind on port␣
 ↪4046. Attempting port 4047.
# opath:  lab2/
# prefix is:  ho
# some lines of original rdd:  ['have\t338996', 'bought\t46988',␣
 ↪'several\t19688', 'of\t792000', 'Vitality\t252']                          ␣
 ↪
# total number of lines in rdd:  339819                                      ␣
 ↪                                                                          ␣
 ↪
# some lines of the filtered rdd:  ['hot\t32944', 'home\t17995',␣
 ↪'however\t12492', 'holds\t1762', 'hot"\t108']
# number of lines in filtered rdd:  1519
# filtered_by_prefix_rdd:  ['hot\t32944', 'home\t17995', 'however\t12492',␣
 ↪'holds\t1762', 'hot"\t108']
# frequencies_rdd:  1519
# max freq:  36264
# number of elements found greater than 0.8*maxfreq:  2
```

[29]:
```
#Question: Take note of the time that it takes to run the applications in the␣
 ↪two different submission modes. Is
#there a difference in time? Can you give a plausible explanation?
# yarn ~ 22 seconds
# local ~ 10 seconds
# The difference in time can probably be attributed to how the data is stored␣
 ↪and operated on.
# using yarn means using the cluster and such means that the code is run in␣
 ↪parallel on the different bloks in different nodes
# using the local means the data is all stored in one place so the code is run␣
 ↪only on that part which consumes less time
```

[ ]:

Bonus

August 7, 2020

[66]:
```
# Bonus Task
```

[67]:
```
rdd = sc.textFile("/data/students/bigdata_internet/lab2/finefoods_text.txt")
```

[68]:
```
print('numer of objects in the input rdd: ',rdd.count())
```

```
numer of objects in the input rdd:  568454
```

[69]:
```
print('Some of the objects in the input rdd: ',rdd.take(2))
```

```
Some of the objects in the input rdd:  [' I have bought several of the Vitality
canned dog food products and have found them all to be of good quality  The
product looks more like a stew than a processed meat and it smells better  My
Labrador is finicky and she appreciates this product better than  most ', '
Product arrived labeled as Jumbo Salted Peanuts   the peanuts were actually
small sized unsalted  Not sure if this was an error or if the vendor intended to
represent the product as "Jumbo" ']
```

[70]:
```
# function that takes a paragraph and gives back a list of single words
def getwords(paragraph):
    words = []
    excludables = ["","  ","-","--","/"]
    for word in paragraph.split(" "):
        if word in excludables:
            continue
        words.append(word)
    return words
```

[71]:
```
# Using flatMap since the final result is the concatenation of the list of␣
 ↪values obtained
# by applying the function f (getwords) over all the elements of the input rdd
words_rdd = rdd.flatMap(getwords)
```

[72]:
```
print('Some of the elements in the words_rdd after the flatMap operation:␣
 ↪',words_rdd.take(60))
```

```
Some of the elements in the words_rdd after the flatMap operation:  ['I',
'have', 'bought', 'several', 'of', 'the', 'Vitality', 'canned', 'dog', 'food',
```

```
'products', 'and', 'have', 'found', 'them', 'all', 'to', 'be', 'of', 'good',
'quality', 'The', 'product', 'looks', 'more', 'like', 'a', 'stew', 'than', 'a',
'processed', 'meat', 'and', 'it', 'smells', 'better', 'My', 'Labrador', 'is',
'finicky', 'and', 'she', 'appreciates', 'this', 'product', 'better', 'than',
'most', 'Product', 'arrived', 'labeled', 'as', 'Jumbo', 'Salted', 'Peanuts',
'the', 'peanuts', 'were', 'actually', 'small']
```

[73]:
```
print('Total number of words in original file: ',words_rdd.count())
```

```
Total number of words in original file:  45347537
```

[74]:
```
# Can you write the code to obtain the word frequency file starting from the␣
 ↪original file?
# Compare the number of words in the file you created and the ones in the word␣
 ↪frequency file
```

[75]:
```
# function to create key, value pairs in order to then be able to apply the␣
 ↪refuceByKey operation
def create_kv_pairs(word):
    return (word,1)
```

[76]:
```
kv_words = words_rdd.map(create_kv_pairs)
```

[77]:
```
print('Some elements of the kv_rdd after being associated word,value:␣
 ↪',kv_words.take(7))
```

```
Some elements of the kv_rdd after being associated word,value:  [('I', 1),
('have', 1), ('bought', 1), ('several', 1), ('of', 1), ('the', 1), ('Vitality',
1)]
```

[78]:
```
# Applying the reduceByKey operation in order to count the total occurances of␣
 ↪each word
word_freq_rdd = kv_words.reduceByKey(lambda accum,n:accum+n)
```

[79]:
```
print('Some of the words and their frequencies: ',word_freq_rdd.take(8))
```

```
Some of the words and their frequencies:  [('like', 244250), ('private', 237),
('reserve', 217), ('ghost', 292), ('2010', 878), ('is', 725903), ('stocked',
1115), ('hot', 32944)]
```

[80]:
```
print('Total number of words in the final rdd: ',word_freq_rdd.count())
```

```
Total number of words in the final rdd:  339817
```

[ ]:
```
# Compared to the given file that has 339819 words the result is very close.
```