Big Data Assignment 1: Word Count with PySpark

Author : Mukesh Ravichandran

CWID : 50380788

Table of Contents

[Part 1: Creating a base RDD and pair RDDs 2](#_Toc195825712)

[(1a) Create a base RDD 2](#_Toc195825713)

[(1b) Pluralize and test 2](#_Toc195825714)

[(1c) Apply makePlural to the base RDD 2](#_Toc195825715)

[(1d) Pass a lambda function to map 3](#_Toc195825716)

[(1e) Length of each word 3](#_Toc195825717)

[(1f) Pair RDDs 3](#_Toc195825718)

[Part 2: Counting with pair RDDs 3](#_Toc195825719)

[(2a) groupByKey() approach 3](#_Toc195825720)

[(2b) Use groupByKey() to obtain the counts 4](#_Toc195825721)

[(2c) Counting using reduceByKey 4](#_Toc195825722)

[(2d) All together 4](#_Toc195825723)

[Part 3: Finding unique words and a mean value 4](#_Toc195825724)

[(3a) Unique words 4](#_Toc195825725)

[(3b) Mean using reduce 4](#_Toc195825726)

[Part 4: Apply word count to a file 5](#_Toc195825727)

[(4a) wordCount function 5](#_Toc195825728)

[(4b) Capitalization and punctuation 5](#_Toc195825729)

[(4c) Load a text file 6](#_Toc195825730)

[(4d) Words from lines 6](#_Toc195825731)

[(4e) Remove empty elements 7](#_Toc195825732)

[(4f) Count the words 7](#_Toc195825733)

# Part 1: Creating a base RDD and pair RDDs

## (1a) Create a base RDD

from pyspark.sql import SparkSession

# Initialize Spark Session

spark = SparkSession.builder.appName("SimplePySpark").getOrCreate()

sc = spark.sparkContext

wordsList = ['cat', 'elephant', 'rat', 'rat', 'cat']

wordsRDD = sc.parallelize(wordsList, 4)

# Print out the type of wordsRDD

print(type(wordsRDD))

## (1b) Pluralize and test

def makePlural(word):

return word+'s'

print( makePlural('cat'))

Findings : the print statement mentioned in the assignment question doesn’t have the brackets, which is needed in the latest version of python

## (1c) Apply makePlural to the base RDD

pluralRDD = wordsRDD.map(makePlural)

print( pluralRDD.collect() )

#['cats', 'elephants', 'rats', 'rats', 'cats']

pluralLambdaRDD = wordsRDD.map(lambda x:x+'s')

print(pluralLambdaRDD.collect())

#'cats', 'elephants', 'rats', 'rats', 'cats']

## (1d) Pass a lambda function to map

pluralLambdaRDD = wordsRDD.map(lambda x:x+'s')

print(pluralLambdaRDD.collect())

## (1e) Length of each word

pluralLengths = pluralRDD.map(lambda x:len(x))

length=pluralLengths.collect()

## (1f) Pair RDDs

wordPairs = wordsRDD.map(lambda x:(x,1))

print ('debug start')

print (wordPairs.collect())

#[('cat', 1), ('elephant', 1), ('rat', 1), ('rat', 1), ('cat', 1)]

# Part 2: Counting with pair RDDs

## (2a) groupByKey() approach

ordsGrouped = wordPairs.groupByKey()

for key, value in wordsGrouped.collect():

print('{0}: {1}'.format(key, list(value)))

# rat: [1, 1]

# elephant: [1]

# cat: [1, 1]

print (sorted(wordsGrouped.mapValues(lambda x: list(x)).collect()))

#[('cat', [1, 1]), ('elephant', [1]), ('rat', [1, 1])]

Findings : Here the sorted has to be printed, this is missing in the assignment

## (2b) Use groupByKey() to obtain the counts

def foo(bar):

print (bar)

print (len (bar))

return 'k'

wordCountsGrouped = wordsGrouped.map(lambda x : (x[0], sum( x[1])))

print (wordCountsGrouped.collect())

#[('rat', 2), ('elephant', 1), ('cat', 2)]

## (2c) Counting using reduceByKey

wordCounts = wordPairs.reduceByKey(lambda x,y:x+y)

#print (wordCounts.collect())

#[('rat', 2), ('elephant', 1), ('cat', 2)]

## (2d) All together

wordCountsCollected = (wordsRDD.map(lambda x:(x,1)).reduceByKey(lambda x,y:x+y).collect())

print (wordCountsCollected)

#[('rat', 2), ('elephant', 1), ('cat', 2)]

print(sorted(wordCountsCollected))

#[('cat', 2), ('elephant', 1), ('rat', 2)]

# Part 3: Finding unique words and a mean value

## (3a) Unique words

uniqueWords = wordsRDD.map(lambda x:(x,1)).reduceByKey(lambda x,y:x+y).count()

print (uniqueWords)

## (3b) Mean using reduce

from operator import add

totalCount = (wordCounts.map(lambda x:x[1]).reduce(add))

average = totalCount/float(uniqueWords)

print (totalCount)

#5

print (round(average, 2))

#1.67

# Part 4: Apply word count to a file

## (4a) wordCount function

def wordCount(wordListRDD):

# """Creates a pair RDD with word counts from an RDD of words.

# Args:

# wordListRDD (RDD of str): An RDD consisting of words.

# Returns:

# RDD of (str, int): An RDD consisting of (word, count) tuples.

# """

return (wordListRDD.map(lambda x:(x,1)).reduceByKey(lambda x,y:x+y))

print (wordCount(wordsRDD).collect())

#[(‘rat', 2), ('elephant', 1), ('cat', 2)]

## (4b) Capitalization and punctuation

import re

def removePunctuation(text):

# """Removes punctuation, changes to lower case, and strips leading and trailing spaces.

# Note:

# Only spaces, letters, and numbers should be retained. Other characters should be

# eliminated (e.g. it's becomes its). Leading and trailing spaces should be removed after

# punctuation is removed.

# Args:

# text (str): A string.

# Returns:

# str: The cleaned up string.

# """

step1 = re.sub(r'[^\w\s]','',text)

step2 = re.sub(r'\_','',step1)

lowercase=step2.lower()

strip=lowercase.strip()

return strip

print (removePunctuation('Hi, you!'))

#hi you

print (removePunctuation(' No under\_score!'))

#no underscore

print (removePunctuation(' \* Remove punctuation then spaces \* '))

## (4c) Load a text file

fileName = "shakespeare.txt"

shakespeareRDD = (sc

.textFile(fileName, 8)

.map(removePunctuation))

print ('\n'.join(shakespeareRDD

.zipWithIndex() # to (line, lineNum)

.map(lambda x: '{0}: {1}'.format(x[1], x[0])) # to 'lineNum: line'

.take(15)))

## (4d) Words from lines

shakespeareWordsRDD = shakespeareRDD.flatMap(lambda x:x.split(" ") )

shakespeareWordCount = shakespeareWordsRDD.count()

print (shakespeareWordsRDD.top(5))

#[u'zwaggerd', u'zounds', u'zounds', u'zounds', u'zounds']

print (shakespeareWordCount)

#946354

Findings: Not able to get eh “u “ in the top 5 records while it is printed.

## (4e) Remove empty elements

shakeWordsRDD = shakespeareWordsRDD.filter(lambda x:x.strip()!="")

shakeWordCount = shakeWordsRDD.count()

print (shakeWordCount)

#901109

## (4f) Count the words

top15WordsAndCounts = wordCount(shakeWordsRDD).takeOrdered(15,lambda x:-x[1])

print ('\n'.join(map(lambda w: '{0}: {1}'.format(w[0],w[1]), top15WordsAndCounts)))

# the: 27645

# and: 26733

# i: 20683

# to: 19198

# of: 18180

# a: 14613

# you: 13650

# my: 12480

# that: 11122

# in: 10967

# is: 9598

# not: 8725

# for: 8245

# with: 7996

# me: 7768