PI2_Lucas_Citolin

February 9, 2020

[1]:
```
from pyspark import SparkContext
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
import json
import pandas as pd
sc = SparkContext('local', appName='AA')
```

[2]:
```
sc._jsc.hadoopConfiguration().set("textinputformat.record.delimiter",␣
↪"\r\n\r\n")
```

[3]:
```
file = '/home/citolin/Documents/ufam/big-data/p3/data/amazon-meta.txt'
data = sc.textFile(file)
```

0.1 Utils

Parse function transforms the textual document into a python dictionary

[4]:
```
def parse(text):
    lines = text.split('\r\n')
    obj = {}
    ID = -1
    reading_categories = False
    reading_reviews = False
    for i in lines:

        if reading_categories:
            categories = i.split('|')
            if(len(categories) == 1):
                reading_categories = False
            else:
                obj['CATEGORIES']['categories'].
↪append(categories[len(categories)-1])

        if reading_reviews:
            reviewsObj = {}
```

```
            reviewsObj['date'] = i.split(' cutomer')[0].replace(' ', '')
            reviews = i.split(': ')
            reviewsObj['customer'] = reviews[1].split(
                ' rating')[0].replace(' ', '')
            reviewsObj['rating'] = reviews[2].split(' ')[0].replace(' ', '')
            reviewsObj['votes'] = reviews[3].split(
                ' helpful')[0].replace(' ', '')
            reviewsObj['helpful'] = reviews[4].replace(' ', '')
            obj['REVIEWS']['reviews'].append(reviewsObj)

        elif 'Id:' in i:
            obj['ID'] = i.split('Id:')[1].replace(' ', '')
            ID = obj['ID']
        elif 'ASIN:' in i:
            obj['ASIN'] = i.split('ASIN:')[1].replace(' ', '')
        elif 'title:' in i:
            title = i.split('title: ')[1]
            obj['TITLE'] = title
        elif 'group:' in i:
            obj['GROUP'] = i.split('group: ')[1]
        elif 'salesrank:' in i:
            obj['SALESRANK'] = i.split('salesrank: ')[1]
        elif 'similar:' in i:
            similars = i.split(': ')[1].split('  ')
            number_of_similars = similars[0]
            similars.pop(0)
            obj['SIMILARS'] = {
                'size': number_of_similars, 'similars': similars}
        elif 'categories:' in i:
            number_of_categories = i.split(': ')[1]
            obj['CATEGORIES'] = {
                'size': number_of_categories, 'categories': []}
            reading_categories = True
        if 'reviews:' in i:
            reviews_split = i.split(': ')
            total = reviews_split[2].split('  ')[0]
            downloaded = reviews_split[3].split('  ')[0]
            avg_rating = reviews_split[4]
            obj['REVIEWS'] = {'total': total, 'downloaded': downloaded,
                              'avg_rating': avg_rating, 'reviews': []}
            reading_reviews = True
    return obj


# Doc = Text Document
def filterById(doc, selected_id):
    lines = doc.split('\n')
```

```
    for line in lines:
        if 'Id:  ' in line:
            ID = int(line.split('Id:')[1].replace(' ', ''))
            if( ID == selected_id ):
                return True
            else:
                return None

# Just to print
class bcolors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'
```

0.2 Questão A

0.2.1 Dado um produto, listar:

0.2.2 — Os 5 comentários mais úteis e com a maior avaliação.

0.2.3 — Os 5 comentários mais úteis e com a menor avaliação.

[5]:
```
def getReviewsByUseful(doc):
    obj = parse(doc)
    try:
        array = []
        for review in obj['REVIEWS']['reviews']:
            rev = ( review['date'], review['customer'], int(review['rating']),␣
 ↪int(review['votes']), int(review['helpful']) )
            array.append( ( int(review['helpful']), ( int(review['rating']),␣
 ↪rev ) ) )
        return array
    except Exception as e:
        return array

# MACRO's for
SELECTED_ID = 2
KEY = 0
VALUE = 1
RATING = 0
```

```
REVIEW = 1

most_useful = data.filter(lambda x: filterById(x, SELECTED_ID)) \
            .flatMap(getReviewsByUseful) \
            .sortByKey(ascending = False) \
            .take(5)

rdd_most_useful = sc.parallelize(most_useful)

best_rating = rdd_most_useful.sortBy( lambda x: x[VALUE][RATING], ascending =␣
↪False ) \
                             .map( lambda x: x[VALUE] ) \
                             .values() \
                             .collect()
worst_rating = rdd_most_useful.sortBy( lambda x: x[VALUE][RATING], ascending =␣
↪True ) \
                             .map( lambda x: x[VALUE] ) \
                             .values() \
                             .collect()
```

0.2.4 5 comentários mais úteis ordenados pela MAIOR avaliação

[6]:
```
df = pd.DataFrame(data=best_rating, columns=['Date', 'Customer', 'Rating',␣
↪'Votes', 'Helpful'])
df
```

[6]:

	Date	Customer	Rating	Votes	Helpful
0	2002-1-24	A13SG9ACZ9O5IM	5	8	8
1	2002-5-23	A1GIL64QK68WKL	5	8	8
2	2002-2-6	A2P6KAWXJ16234	4	16	16
3	2002-3-23	A3GO7UV9XX14D8	4	6	6
4	2004-2-11	A1CP26N8RHYVVO	1	13	9

0.2.5 5 comentários mais úteis ordenados pela MENOR avaliação

[7]:
```
df = pd.DataFrame(data=worst_rating, columns=['Date', 'Customer', 'Rating',␣
↪'Votes', 'Helpful'])
df
```

[7]:

	Date	Customer	Rating	Votes	Helpful
0	2004-2-11	A1CP26N8RHYVVO	1	13	9
1	2002-2-6	A2P6KAWXJ16234	4	16	16
2	2002-3-23	A3GO7UV9XX14D8	4	6	6
3	2002-1-24	A13SG9ACZ9O5IM	5	8	8

```
4   2002-5-23   A1GIL64QK68WKL       5       8         8
```

0.3 Questão B

0.3.1 Dado um produto, listar os produtos similares com maiores vendas do que ele

[8]:
```
SELECTED_ID = 2

# Filter the required ID
# Map emits a ( ID, Text Doc ) tuple
rdd_id = data.filter(lambda x: filterById(x, SELECTED_ID))

# Convert the Textual Document into a Dictionary to extract the SIMILARS and␣
 ↪the SALESRANK
obj = parse(rdd_id.first())
similars = [ int(similar) for similar in obj['SIMILARS']['similars'] ]
minimal_salesrank = obj['SALESRANK']

# In map filter
#    Checks if ASIN is in the similar list
#    Checks if the SALESRANK is lower than the minimal_salesrank
def filterSimilarsBySalesrank(doc, similars, minimal_salesrank):
    obj = parse(doc)
    try:
        if int(obj['ASIN']) in similars and int(obj['SALESRANK']) <=␣
 ↪int(minimal_salesrank):
            return True
        else:
            return None
    except Exception as e:
        return None

# Emit: ( SALESRANK, TITLE )
def emitTitleAndSalesrank(doc):
    obj = parse(doc)
    try:
        return ( int(obj['SALESRANK']), obj['TITLE'] )
    except Exception as e:
        return ( -1, -1 )

# In map filter - Filter similars with smaller salesrank
rdd_b = data.filter(lambda x: filterSimilarsBySalesrank( x, similars,␣
 ↪minimal_salesrank ) ) \
            .map( emitTitleAndSalesrank ) \
            .filter( lambda x: x[KEY] != -1 ) \
```

```
            .sortByKey(ascending=True)

answer_a = rdd_b.collect()
```

[9]:
```
df = pd.DataFrame(data=answer_a, columns=['Salesrank', 'Title'])
df
```

[9]:

	Salesrank	Title
0	58836	Lammas
1	103012	Yule: A Celebration of Light and Warmth
2	159277	Midsummer: Magical Celebrations of the Summer …

0.4 Questão C

0.4.1 Dado um produto, mostrar a evolução diária das médias de avaliação ao longo do intervalo de tempo coberto no arquivo de entrada

[10]:
```
# Um bom exemplo é o ID 19367 (vários reviews)
SELECTED_ID = 19367

def emitReviewsByDate(doc):
    obj = parse(doc)
    array = []
    try:
        avg_rating = obj['REVIEWS']['avg_rating']
        for review in obj['REVIEWS']['reviews']:
            date = datetime.datetime.strptime(review['date'], '%Y-%m-%d').date()
            rating = review['rating']
            array.append( ( date , ( float(rating), float(avg_rating), 1 ) ) )
        return array
    except Exception as e:
        return array

KEY = 0
VALUE = 1
RATING = 0
AVG_RATING = 1


# Filter the required ID (emits a DOC)
# Map emits a ( ID, Text Doc ) tuple
rdd_c = data.filter(lambda x: filterById(x, SELECTED_ID)) \
                .flatMap( emitReviewsByDate ) \
                .sortByKey(ascending=True)
```

```
answer_c = rdd_c.map( lambda x: ( x[KEY], x[VALUE][RATING],␣
↪x[VALUE][AVG_RATING] ) ).collect()
```

[11]:
```
# Get growth by date

## To each date, calculate the mean at the time (instant mean)
def addToAverage(old_av, value, new_size):
    new_av = float(old_av) + ( float(value) - float(old_av) ) / float(new_size)
    return new_av

counter = 0
means_array = [0]
for el in answer_c:
    means_array.append(addToAverage( means_array[ len(means_array)-1 ], el[1],␣
↪len(means_array) ))
means_array.pop(0)
```

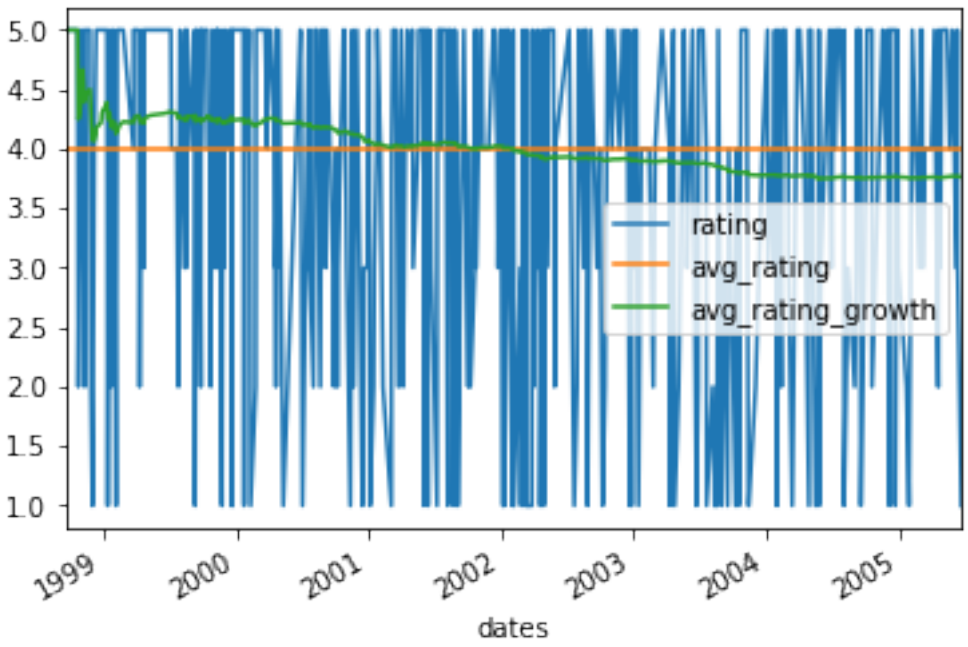
[11]: 0

[12]:
```
df = pd.DataFrame(data=answer_c, columns=['dates', 'rating', 'avg_rating'])
df['avg_rating_growth'] = means_array
df['dates'] = pd.to_datetime(df['dates'],infer_datetime_format=True)
df.plot(x='dates', y=['rating', 'avg_rating', 'avg_rating_growth'])
print(df)
```

```
         dates  rating  avg_rating  avg_rating_growth
0   1998-09-21     5.0         4.0           5.000000
1   1998-10-15     5.0         4.0           5.000000
2   1998-10-21     5.0         4.0           5.000000
3   1998-10-23     2.0         4.0           4.250000
4   1998-11-01     5.0         4.0           4.400000
5   1998-11-02     5.0         4.0           4.500000
6   1998-11-02     5.0         4.0           4.571429
7   1998-11-02     5.0         4.0           4.625000
8   1998-11-04     5.0         4.0           4.666667
9   1998-11-10     2.0         4.0           4.400000
10  1998-11-11     5.0         4.0           4.454545
11  1998-11-24     5.0         4.0           4.500000
12  1998-11-27     2.0         4.0           4.307692
13  1998-12-01     1.0         4.0           4.071429
14  1998-12-06     4.0         4.0           4.066667
15  1998-12-11     5.0         4.0           4.125000
16  1998-12-11     5.0         4.0           4.176471
17  1998-12-25     5.0         4.0           4.222222
18  1998-12-26     5.0         4.0           4.263158
19  1998-12-27     5.0         4.0           4.300000
20  1998-12-31     5.0         4.0           4.333333
```

```
21  1999-01-10     5.0         4.0           4.363636
22  1999-01-10     5.0         4.0           4.391304
23  1999-01-12     1.0         4.0           4.250000
24  1999-01-17     5.0         4.0           4.280000
25  1999-01-17     5.0         4.0           4.307692
26  1999-01-21     2.0         4.0           4.222222
27  1999-01-23     3.0         4.0           4.178571
28  1999-01-26     5.0         4.0           4.206897
29  1999-01-28     5.0         4.0           4.233333
..         …     …         …               …
419 2004-12-19     5.0         4.0           3.766667
420 2004-12-22     5.0         4.0           3.769596
421 2004-12-22     1.0         4.0           3.763033
422 2004-12-28     5.0         4.0           3.765957
423 2004-12-31     3.0         4.0           3.764151
424 2005-01-22     2.0         4.0           3.760000
425 2005-01-29     1.0         4.0           3.753521
426 2005-02-07     5.0         4.0           3.756440
427 2005-02-26     3.0         4.0           3.754673
428 2005-03-01     5.0         4.0           3.757576
429 2005-03-06     5.0         4.0           3.760465
430 2005-03-08     3.0         4.0           3.758701
431 2005-03-12     4.0         4.0           3.759259
432 2005-03-28     4.0         4.0           3.759815
433 2005-04-01     4.0         4.0           3.760369
434 2005-04-09     5.0         4.0           3.763218
435 2005-04-19     2.0         4.0           3.759174
436 2005-04-20     2.0         4.0           3.755149
437 2005-04-21     5.0         4.0           3.757991
438 2005-04-25     3.0         4.0           3.756264
439 2005-04-29     5.0         4.0           3.759091
440 2005-05-07     5.0         4.0           3.761905
441 2005-05-14     5.0         4.0           3.764706
442 2005-05-21     4.0         4.0           3.765237
443 2005-05-29     5.0         4.0           3.768018
444 2005-05-29     5.0         4.0           3.770787
445 2005-06-08     3.0         4.0           3.769058
446 2005-06-11     5.0         4.0           3.771812
447 2005-06-12     3.0         4.0           3.770089
448 2005-06-22     1.0         4.0           3.763920

[449 rows x 4 columns]
```



0.5 Questão D

0.5.1 Listar os 10 produtos lideres de venda em cada grupo de produto

[13]:
```
def emitByGroupNew(doc):
    obj = parse(doc)
    try:
        group = obj['GROUP']
        salesrank = obj['SALESRANK']
        product_id = obj['ID']
        title = obj['TITLE']
        return ( group, ( int(salesrank), title ) )
    except Exception as e:
        return ( -1, (-1, -1, -1) )

KEY = 0
VALUE = 1
SALESRANK = 0
TITLE = 1

rdd_d = data.map( emitByGroupNew ) \
            .filter( lambda x: x[VALUE][SALESRANK] != -1 ) \
            .sortBy( lambda x: x[VALUE][SALESRANK], ascending=True ) \
            .groupByKey() \
```

```
            .map( lambda x: ( x[KEY], list(x[VALUE])))

answer_d = rdd_d.collect()
```

[14]:
```
for group in answer_d:
    print(bcolors.BOLD + '>>> Group: ' + str(group[0]) + bcolors.ENDC)
    count = 0
    for el in group[1]:
        print(bcolors.OKGREEN + '\tSalesrank: ' + str( el[0] ) + bcolors.ENDC +␣
↪'\t' + el[1] )
        count = count + 1
        if(count == 10):
            break
```

```
>>> Group: Video
   Salesrank: 0         From Soup to Nuts
   Salesrank: 1         The War of the Worlds
   Salesrank: 2         Shirley Valentine
   Salesrank: 6         Leslie Sansone - Walk Away the Pounds - Super
Fat Burning
   Salesrank: 7         Robin Hood - Men in Tights
   Salesrank: 8         Richard Simmons - Sweatin' to the Oldies
   Salesrank: 12        Howard the Duck
   Salesrank: 14        Charlotte's Web
   Salesrank: 16        A Tree Grows in Brooklyn
   Salesrank: 17        My Neighbor Totoro
>>> Group: Music
   Salesrank: 0         Improvisations - Jazz In Paris
   Salesrank: 0         Lucky Man
   Salesrank: 0         I Need Your Loving
   Salesrank: 0         That Travelin' Two-Beat/Sings the Great Country
Hits
   Salesrank: 27        Buzz Buzz
   Salesrank: 33        A Rush of Blood to the Head
   Salesrank: 42        Michael Bublé
   Salesrank: 46        Come Away with Me
   Salesrank: 53        Songs About Jane
   Salesrank: 55        Facing Future
>>> Group: Sports
   Salesrank: 4684      Yoga Kit Living Arts
>>> Group: DVD
   Salesrank: 0         The Drifter
   Salesrank: 0         The House Of Morecock
   Salesrank: 0         1, 2, 3 Soleils: Taha, Khaled, Faudel
   Salesrank: 28        Star Wars - Episode I, The Phantom Menace
(Widescreen Edition)
   Salesrank: 47        Band of Brothers
```

```
   Salesrank: 49       The Little Mermaid (Limited Issue)
   Salesrank: 55       The Wizard of Oz
   Salesrank: 85       Fawlty Towers - The Complete Collection
   Salesrank: 85       Star Wars - Episode II, Attack of the Clones
(Widescreen Edition)
   Salesrank: 88       Jerry Seinfeld Live on Broadway: I'm Telling You
for the Last Time
>>> Group: Video Games
   Salesrank: 339      PRIMA PUBLISHING Dark Cloud 2 Official Strategy
Guide
>>> Group: Book
   Salesrank: 0        How to Start and Run Your Own Mystery Shopping
Company
   Salesrank: 0        Common Sense About Uncommon Wisdom: Ancient
Teachings of Vedanta
   Salesrank: 0        How To Get The Best Creative Work From Your
Agency: Advertising, Interactive And Other Marketing Communications
   Salesrank: 0        Help Me Talk Right: How to Teach a Child to Say
the "R" Sound in 15 Easy Lessons
   Salesrank: 0        Gods on Earth (Thor, Book 3)
   Salesrank: 0        El arte de amar
   Salesrank: 0        The Manager's Bible: A Practical Guide for the
Current and Future Manager
   Salesrank: 0        Dona Barbara
   Salesrank: 0        The Irish Americans: The Immigrant Experience
   Salesrank: 0        The Diligent: A Voyage through the Worlds of the
Slave Trade
>>> Group: Toy
   Salesrank: 59       IlluStory Book Kit
   Salesrank: 1890     Wizard Card Game Deluxe
   Salesrank: 2288     Photostory Junior Book Kit
   Salesrank: 4053     Party Tyme Karaoke CD Oldies
   Salesrank: 7812     Party Tyme Karaoke CD Kids Songs
   Salesrank: 10732    Party Tyme Karaoke CD: V2 Super Hits
   Salesrank: 31296    The Songs of Britney Spears & Christina Aguilera
   Salesrank: 45241    R- Photostory Senior
>>> Group: Baby Product
   Salesrank: 1017     Baby'S Record Keeper And Memory Box
>>> Group: Software
   Salesrank: 200      ClickArt Christian Publishing Suite 3
   Salesrank: 327      RINGDISC Wagner: The Ring Disc
   Salesrank: 1955     Zondervan Bible Study Library: Leader's Edition
5.0
   Salesrank: 3771     Just Enough Vocals The Learning Co
   Salesrank: 3828     WINDOWS NT SERVER V4.0 RESOURCE
>>> Group: CE
   Salesrank: 39367    SPELLING CORRECTOR
   Salesrank: 69089    Hal Leonard Beginning Bass Guitar 1,
```

```
Instructional Video, 30 Minutes
   Salesrank: 71678    Hal Leonard Beginning Guitar 1, Instructional
Video, 30 Minutes
   Salesrank: 84976    FRANKLIN COMP. KJB-1440 Electronic Holy Bible
(King James Version)
```

0.6 Questão E

0.6.1 Listar os 10 produtos com a maior média de avaliações úteis positivas

[15]:
```
POSITIVE_RATING = 5

def getPositiveReviewsByIdNew(doc):
    try:
        obj = parse(doc)
        array = []
        title = obj['TITLE']
        product_id = obj['ID']
        for review in obj['REVIEWS']['reviews']:
            helpful = review['helpful']
            rating = int(review['rating'])
            if rating >= POSITIVE_RATING:
                array.append( ( int(product_id), ( int(helpful), 1, title ) ) )
        return array
    except Exception as e:
        return [ ( -1, (-1,-1,-1) ) ]

KEY = 0
VALUE = 1

HELPFUL = 0
COUNTER = 1
TITLE = 2
HELPFUL_AVERAGE = 0

# Already filter in-map
rdd_e = data.flatMap(getPositiveReviewsByIdNew) \
            .filter(lambda x: x[KEY] != -1 ) \
            .reduceByKey(lambda x, y: (x[HELPFUL] + y[HELPFUL], x[COUNTER] +␣
 ↪y[COUNTER], x[TITLE])) \
            .map(lambda x: (x[VALUE][HELPFUL]/x[VALUE][COUNTER],␣
 ↪x[VALUE][TITLE])) \
            .sortBy(lambda x: x[HELPFUL_AVERAGE], ascending=False) \

answer_e = rdd_e.take(10)
```

[16]:
```
df = pd.DataFrame(data=answer_e, columns=['Helpful', 'Title'])
df
```

[16]:

	Helpful	Title
0	320	Easy Adult Piano Beginner's Course
1	247	Small Engine Repair Up to 20 Hp
2	233	T'ai Chi for Older Adults
3	231	The Story About Ping (8x 8)
4	231	The Story about Ping : StoryTape (StoryTape, P…
5	231	The Story about Ping
6	231	The Story About Ping
7	206	The Glucose Revolution Pocket Guide to the Top…
8	203	More Than Just Hot Air: Common Sense Counter-T…
9	197	Crockpot Cookery (Cookbooks By Morris Press)

0.7 Questão F

0.7.1 Listar as 5 categorias de produto com a maior média de avaliações úteis positivas por produto

[17]:
```
# WITH OBJECT
POSITIVE_RATING = 5

def getPositiveReviewsByCategory(doc):
    obj = parse(doc)
    array = []
    try:
        review_array = [ ( int(r['helpful']), 1) for r in␣
 ↪obj['REVIEWS']['reviews'] if int(r['rating']) >= POSITIVE_RATING ]
        for el in obj['CATEGORIES']['categories']:
            for review in review_array:
                array.append( ( el, review ) )
        return array
    except Exception as e:
        return array

KEY = 0
VALUE = 1

HELPFUL = 0
COUNTER = 1

# In map filter
rdd_f = data.flatMap( getPositiveReviewsByCategory ) \
            .reduceByKey(lambda x, y: ( x[HELPFUL] + y[HELPFUL], x[COUNTER] +␣
 ↪y[COUNTER] ) ) \
```

```
            .mapValues(lambda x: ( x[HELPFUL]/x[COUNTER] ) ) \
            .sortBy(lambda x: x[VALUE], ascending=False)

answer_f = rdd_f.take(5)
```

[18]:
```
df = pd.DataFrame(data=answer_f, columns=['Title', 'Rating mean'] )
df
```

[18]:

	Title	Rating mean
0	Dominican Republic[4816]	97
1	Casual Users[502030]	84
2	Casual Users[727732]	84
3	Adventures of Ozzie and Harriet[13744851]	80
4	Jack Benny Program[13744881]	80

0.8 Questão G

0.8.1 Listar os 10 clientes que mais fizeram comentários por grupo de produtos

[19]:
```
def getCustomerReviewsByGroup(doc):
    obj = parse(doc)
    array = []
    try:
        for review in obj['REVIEWS']['reviews']:
            array.append( ( ( obj['GROUP'], review['customer'] ) , 1  ) )
        return array
    except Exception as e:
        return array

KEY   = 0
VALUE = 1

GROUP = 0
CUSTOMER = 1
COUNT = 1

rdd_g = data.flatMap( getCustomerReviewsByGroup ) \
            .reduceByKey( lambda x,y: x + y ) \
            .map( lambda x: ( x[KEY][GROUP], ( x[KEY][CUSTOMER], x[VALUE] ) ) )␣
 ↪\
            .sortBy( lambda x: ( x[KEY], x[VALUE][COUNT] ), ascending = False )␣
 ↪\
            .groupByKey() \
            .map( lambda x: (x[0], list(x[1])))
```

```
answer_g = rdd_g.collect()
```

[20]:
```
for group_tuple in answer_g:
    count = 0
    print(bcolors.BOLD + '>> Group: ' + group_tuple[0] + bcolors.ENDC)
    for customer in group_tuple[1]:
        print('\t\t ' + customer[0] + '\t' + bcolors.OKGREEN + str(customer[1])␣
↪+ bcolors.ENDC)
        count = count + 1
        if(count == 10):
            break
```

```
>> Group: Video
                A1VPDKIKX0DER  72581
                A3UN6WX5RRO2AG 15814
                A2NJO6YE954DBH 1775
                AU8552YCOO5QX  1205
                A3P1A63Q8L32C5 737
                A2OEEWWSFMZ1PN 720
                A16CZRQL23NOIW 668
                A3LZGLA88K0LAO 614
                A2QRB6L1MCJ53G 606
                A152C8GYY25HAH 583
>> Group: Music
                A1VPDKIKX0DER  166149
                A3UN6WX5RRO2AG 15875
                A9Q28YTLYREO7  2760
                A2U49LUUY4IKQQ 1258
                A1GN8UJIZLCA59 1154
                A2NJO6YE954DBH 1128
                A1J5KCZC8CMW9I 1031
                A3MOF5KF93Q6WE 989
                AXFI7TAWD6H6X  814
                A38U2M9OAEJAXJ 780
>> Group: Sports
                A1W18OY9O1FALI 1
                A18ZVYTEDAOF9A 1
                A3O8EZOX2P399L 1
                AL62LOJKDES3M  1
                A2RHSQZ7MAKKCO 1
>> Group: DVD
                A1VPDKIKX0DER  63148
                A3UN6WX5RRO2AG 15549
                A2NJO6YE954DBH 1366
                AU8552YCOO5QX  1213
                A3P1A63Q8L32C5 859
                A3LZGLA88K0LAO 856
```

```
                A82LIVYSX6WZ9  683
                A152C8GYY25HAH 675
                A16CZRQL23NOIW 651
                A1CZICCYP2M5PX 650
>> Group: Video Games
                A226EDS7WDF7S1 1
                A3C811U31YG6FS 1
                A1M4NJYP0WNL8Q 1
>> Group: Book
                ATVPDKIKX0DER  643185
                A3UN6WX5RRO2AG 154531
                A14OJS0VWMOSWO 9589
                AFVQZQ8PW0L    5441
                A1K1JW1C5CUSUZ 3562
                A2NJO6YE954DBH 2055
                A3QVAKVRAH657N 1651
                A1NATT3PN24QWY 1535
                A1D2C0WDCSHUWZ 1508
                A2ODBHT4URXVXQ 1469
>> Group: Toy
                AH4M07U4YC695  2
                ATVPDKIKX0DER  2
                A1SB7SB31ETYZH 2
                A1OA2ZW406NQXM 1
                A1QW8PHDJBH4IC 1
                A1FPVUL053AKXO 1
                A34KPXP8CGUBO5 1
                AQJ2XVMHXGN9A  1
                A1ZO50XHPV0QPQ 1
                AXNOIMARQCT3M  1
>> Group: Baby Product
                A2LAH8VX720175 1
                AI9SB5VKUFXDC  1
                A37TFIP0OMKGMW 1
>> Group: Software
                A3D5ICIQ8STPCH 1
                A1EIVBXG3RD15O 1
                A36NHJPD24UMGJ 1
                A36T3O4TIC1YDQ 1
                A1F8RIBFWRYM3Y 1
                A2I0ZWBVRO575O 1
                AK9MWTH6LJF64  1
                A37UFPGDSSMEV  1
                A1XQB8IU7S8WEU 1
                A183K8JAQJW8LZ 1
>> Group: CE
                A13JU90C7AU3RT 1
                A2IX9TMXDBUCYV 1
```

```
A1SFX3CR838F36 1
A1328SYT22GA4U 1
A1J62O1S6QTHZJ 1
```