



به نام خداوند بخشنده و مهربان

تمرین اول: مقدمه‌ای بر اسپارک

درس: پایگاه داده پیشرفته

استاد: محمدعلی نعمت‌بخش

دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام خانوادگی: سارا معینی

آدرس گیت:

در این تمرین، هدف ما آشنایی با Action و Transformation در موتور تحلیلی Spark است.

۱. منظور از Lazy Evaluation در Spark چیست؟ این مفهوم را همراه با یک مثال توضیح دهید.

این اصطلاح به این معناست که عملیات بلافاصله اجرا نمی‌شوند بلکه spark عملیات‌های اعمال شده را از طریق یک گراف بدون دور حفظ می‌کند در واقع به این معنی است که تا زمانی که یک عمل فعال نشود، اجرا شروع نخواهد شد. به عنوان مثال فرض کنید به ما گفته می‌شود شی‌ای را خریداری کنیم. بعد از خریداری دوباره گفته می‌شود که شی‌ای دیگر خریداری شود و ما مجبور می‌شویم دوباره به مغازه برویم. ولی اگر صبر می‌کردیم و لیست خریدی تهیه می‌کردیم و همه را با هم می‌خریدیم در زمان صرفه جویی میشد. به طور مشابه، اگر Spark بتواند تا فراخوانی یک action صبر کند، ممکن است برخی از تبدیل‌ها را ادغام کند یا به طور کامل از برخی تبدیل‌های غیر ضروری چشم‌پوشی کند و یک برنامه اجرایی کامل آماده کند.

۲. منظور از Wide Transmittion (WT) و Narrow Transmittaion (NT) را در Spark همراه با یک مثال بیان کنید. تفاوت اصلی این دو مفهوم چیست؟

هنگامی که هر پارتیشن در RDD والد حداکثر توسط یک پارتیشن از RDD فرزند استفاده می‌شود، آنگاه یک Narrow transformations (NT) خواهیم داشت. با این تبدیل، Spark به طور خودکار عملیاتی به نام pipelining را انجام می‌دهد، به این معنی که اگر چندین فیلتر را روی DataFrames تعیین کنیم، همه آنها در حافظه انجام می‌شوند. مانند filter()

Wide transformations نتیجه groupByKey و reduceByKey هستند. همه عناصری که برای محاسبه رکوردها در یک پارتیشن مورد نیاز هستند ممکن است در بسیاری از پارتیشن‌های RDD والد زندگی کنند.

با وابستگی گسترده، هر پارتیشن فرزند به هر پارتیشن والدین خود بستگی دارد. این رابطه چند به چند است. با وابستگی محدود، هر پارتیشن فرزند حداکثر به یک پارتیشن از هر والدین بستگی دارد. این رابطه می‌تواند

رابطه یک به یک یا چند به یک با شد. در مقایسه با وابستگی باریک، وابستگی گسترده به دلیل shuffling، عملیات گران‌تری هستند. همچنین سرعت وابستگی باریک بیشتر از وابستگی گسترده است.

۳. با توجه به سوال پیشین، ۴ مورد از NT، WT و Action هایی که در اسپارک وجود دارند نام ببرید.

NT: map(), mapPartition(), flatMap(), filter(), union()

Action: first(), min(), max(), foreach()

Wt: groupByKey() , aggregateByKey() , aggregate() , join()

۴. برای آشنایی بیشتر با مفاهیم بیان شده و مقدمه‌ای بر توابع عملیات‌های زیر را انجام داده و خروجی هریک به همراه بلاک کد آن را گزارش دهید. مثالی از خروجی برای هر بخش نمایش داده شده است.

- برای کار با اسپارک، کتابخانه‌ای با نام pyspark وجود دارد.
- نوت‌بوکی بر روی گوگل کولب ایجاد کرده و این کتابخانه را فراخوانی کنید.

```
✓ [1] !pip install pyspark py4j
52s

import pyspark
from pyspark.sql import SparkSession

Collecting pyspark
  Downloading pyspark-3.2.1.tar.gz (281.4 MB)
    |████████████████████████████████████████| 281.4 M
```

- سپس یک لیست ۵۰ تایی از یک موضوع را برای خود در ست کنید. برای مثال لیستی از (کتاب‌ها، نرم‌افزارها و ...)

```
Degrees = ['Computer Science', 'Finance', 'Marketing', 'Chemistry', 'Accounting', 'Graphic Design', 'Anthropology',
'Architecture', 'Biochemistry', 'Biology', 'Biomedical Engineering', 'Biotechnology', 'Civil Engineering', 'Electrical Engineering',
'History', 'Journalism', 'Sports Management', 'Nursing', 'Psychology', 'Political Science', 'Mathematics', 'Physics', 'Food Technology',
'IT', 'Fashion Design', 'Mechanical Engineering', 'Philosophy', 'Interior Design', 'Statistics', 'Sociology', 'Manufacturing Engineering',
'Aeronautical Engineering', 'Aerospace Engineering', 'Retail Management', 'Naturopathy', 'Bachelor Of Surgery', 'law',
'Physiotherapy', 'Business Administration', 'Environmental Science', 'Astronomy', 'Human Resource Management', 'Pharmacy', 'Graphic',
'Occupational Therapy', 'Agricultural Engineering', 'Atmospheric Science', 'Events Management', 'Textile Engineering', 'Petroleum Engineering',
]
```

- لیست خود را به RDD تبدیل کنید.

با استفاده از تابع parallelize از کتابخانه اسپارک همانطور که در ردیف آخر شکل زیر نشان داده شده لیست را به RDD تبدیل می‌کنیم.

```
spark=SparkSession.builder.appName('local').getOrCreate()
```


```
sc=spark.sparkContext
```

```
rdd=sc.parallelize(Degrees)
```

- با کمک دستور filter بر روی RDD، از آن برای بازیابی عنصر ۲۰ام لیست خود استفاده کنید. (برابر با عنصر ۲۰ام باشد)

```
rdd.filter(lambda x: Degrees[19] in x).collect()  
['Political Science']
```

- با کمک map تمامی عناصر لیست خود را به حروف بزرگ تبدیل و آن را بازیابی کنید.

```
 rdd.map(lambda x:x.upper()).collect()
```

```
[  
'COMPUTER SCIENCE'  
'FINANCE'  
'MARKETING'  
'CHEMISTRY'  
'ACCOUNTING'  
'GRAPHIC DESIGN'  
'ANTHROPOLOGY'  
'ARCHITECTURE'  
'BIOCHEMISTRY'  
'BIOLOGY'  
'BIOMEDICAL ENGINEERING'  
'BIOTECHNOLOGY'  
'CIVIL ENGINEERING'  
'ELECTRICAL ENGINEERING'  
'HISTORY'  
'JOURNALISM'  
'SPORTS MANAGEMENT'  
'NURSING'  
'PSYCHOLOGY'  
'POLITICAL SCIENCE'  
'MATHEMATICS'  
'IT'  
'FASHION DESIGN'  
'MECHANICAL ENGINEERING'  
'PHILOSOPHY'  
'INTERIOR DESIGN'  
'STATISTICS'  
'SOCIOLOGY'
```

تابع upper() کار تبدیل به حروف بزرگ را انجام می‌دهد.
بخشی از خروجی به صورت روبه‌رو است.

- با کمک دستور groupby و map، لیست خود را بر اساس اولین کاراکتر آن دسته بندی کنید.

در دستور groupby و با استفاده از x[0] کاراکتر اول هر عنصر را ب دست آورده و با map، عناصر را که با list(x[1]) به دست می‌آید بر طبق کاراکتر گروه بندی می‌کنیم.

```
rdd.groupBy(lambda x:x[0]).map(lambda x : (x[0], list(x[1]))).collect()
```

```
[('C', ['Computer Science', 'Chemistry', 'Civil Engineering']),
 ('J', ['Journalism']),
 ('S', ['Sports Management', 'Statistics', 'Sociology']),
 ('N', ['Nursing', 'Naturopathy']),
 ('R', ['Retail Management']),
 ('L', ['Law']),
 ('O', ['Occupational Therapy']),
 ('F', ['Finance', 'Food Technology', 'Fashion Design']),
 ('M',
 ['Marketing',
 'Mathematics',
 'Mechanical Engineering',
 'Manufacturing Engineering']),
 ('A',
 ['Accounting',
 'Anthropology',
 'Architecture',
 'Aeronautical Engineering',
 'Aerospace Engineering',
 'Astronomy',
 'Agricultural Engineering',
 'Atmospheric Sciences']),
 ('G', ['Graphic Design']),
 ('B',
 ['Biochemistry',
 'Biology',
 'Biomedical Engineering',
 'Biotechnology',
 'Bachelor of Surgery',
 'Business Administration']),
 ('E',
 ['Electrical Engineering', 'Environmental Science', 'Event Management']),
 ('H', ['History', 'Human Resource Management']),
 ('P',
 ['Psychology',
 'Political Science',
 'Physics',
 'Philosophy',
 'Physiotherapy',
 'Pharmacy',
 'Petroleum Engineering']),
 ('I', ['IT', 'Interior Design']),
 ('T', ['Textile Engineering'])]
```

- عملیات map و reduce را بر روی یک متن نسبتاً بلند پس از تبدیل توکن‌های آن به rdd، انجام دهید.

ابتدا فایل داده را میخوانیم. g در قالب rdd میباشد.

```
g=sc.textFile('./sample_data/data.txt')
```

سپس تابع زیر را نوشته که داده را به توکن تبدیل میکند.

```
import string
def tokenize(line):
    table = dict.fromkeys(map(ord, string.punctuation))
    return line.translate(table).lower().split()
```

```
words = g.flatMap(lambda line: tokenize(line))
```

از تابع map() برای انجام عملیات مپ استفاده میکنیم. `words = words.map(lambda x: (x, 1))`
`words.take(10)`

```
[('nature', 1),
 ('refers', 1),
 ('to', 1),
 ('the', 1),
 ('interaction', 1),
 ('between', 1),
 ('the', 1),
 ('physical', 1),
 ('surroundings', 1),
 ('around', 1)]
```

```
counts = words.reduceByKey(lambda x, y: x+y)
counts.take(10)
```

و در آخر مانند روبه رو عملیات reduce را انجام میدهیم.

```
[('refers', 1),
 ('interaction', 4),
 ('around', 2),
 ('us', 9),
 ('like', 6),
 ('climate', 2),
 ('resources', 2),
 ('flora', 1),
 ('is', 19),
 ('indeed', 1)]
```

شکل زیر ترکیب کدهای بالا در یک تکه کد است.

```
(  
g.flatMap(lambda line: tokenize(line))  
    .map(lambda word: (word, 1))  
    .reduceByKey(lambda x, y: x + y)  
    .takeOrdered(40, key=lambda x: -x[1])  
)
```

بخشی از خروجی نهایی:

```
[('the', 42),  
 ('nature', 37),  
 ('of', 30),  
 ('and', 29),  
 ('is', 19),  
 ('in', 14),  
 ('we', 13),  
 ('to', 13),  
 ('our', 10),  
 ('us', 9),  
 ('are', 9),  
 ('that', 9),  
 ('for', 8),  
 ('services', 8),  
 ('natural', 7),  
 ('a', 7),  
 ('like', 6),  
 ('as', 6),  
 ('all', 6),  
 ('from', 6),  
 ('by', 6),  
 ('with', 6),  
 ('earth', 5),  
 ('air', 5),  
 ('interaction', 4),  
 ('water', 4),  
 ('it', 4),  
 ('ecosystem', 4),  
 ('humans', 4),  
 ('on', 4),  
 ('trees', 4),
```

- چه تفاوتی بین Actionهای take و collect وجود دارد؟

دستور collect همه‌ی عناصر و محتوای RDD را برمی‌گرداند ولی با take تعداد محدودی از عناصر را می‌توان دریافت کرد و می‌توان مشخص کرد که چند تا از عناصر نشان داده شوند. مثلاً take(10) ده عنصر اول را نشان می‌دهد.