



به نام خداوند بخشنده و مهربان

تمرین اول: مقدمه‌ای بر اسپارک

استاد: محمدعلی نعمت‌بخش

درس: پایگاه داده پیشرفته

دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام خانوادگی: مهناز توحیدی‌مهر

آدرس گیت: <https://github.com/mtohidimehr/bigData-hw-one.git>

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.

منظور از Lazy Evaluation در اسپارک این است که تا زمانی که یک action فراخوانی نشود، اسپارک اجرای فرآیند را آغاز نمی‌کند. تا زمانی که تنها عملیات transformation انجام می‌شود، اسپارک منتظر می‌ماند. وقتی که یک action فراخوانی شد، اسپارک با توجه به همه‌ی transformation ها دنباله‌ای از اعمالی که برای گرفتن خروجی مورد نیاز است را ایجاد می‌کند. به عنوان مثال یک Dataframe شامل یک ستون به نام cl-one و ۱۰۰۰ سطر داریم حال می‌خواهیم دو سناریو زیر را روی آن اجرا کنیم.

سناریو ۱:

اضافه کردن ستون cl-two به Dataframe

چاپ Dataframe نهایی

سناریو ۲:

اضافه کردن ستون cl-two به Dataframe

حذف کردن ستون cl-two از Dataframe

چاپ Dataframe نهایی

در حالت عادی سناریو دوم به خاطر داشتن یک عملیات اضافه و یک عملیات حذف باید زمان بیشتری نسبت به سناریوی اول لازم داشته باشد. اما به خاطر ویژگی Lazy Evaluation، اسپارک متوجه می‌شود که عملیات اضافه و حذف در سناریوی دوم حاصلی ندارند پس این مراحل را نادیده گرفته و در نتیجه با صرف زمان کمتری نسبت به سناریوی اول کار را تمام می‌کند.

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

Transformation ها به دو دسته تقسیم می‌شوند:

- **Narrow Transformations**: این نوع Transformation هر قسمت از ورودی را به تنها یک قسمت خروجی تبدیل می‌کند. سرعت این نوع Transformation بالاست. به هیچ‌گونه جابجایی داده در شبکه خوشه‌ای نیاز ندارد. عملیات‌های `map()` و `filter()` به این دسته تعلق دارند.
- **Wide Transformations**: این نوع Transformation دارای پارتیشن‌های ورودی است که در بسیاری از پارتیشن‌های خروجی نقش دارند. به نسبت **Narrow Transformations** ها سرعت پایین‌تری دارند زیرا تحت‌تاثیر عملیات `shuffle` بین گره‌های متفاوت در زمان تولید قسمت‌های جدید است. توابعی مثل `groupByKey()`، `aggregateByKey()`، `aggregate()`، `join()`، `repartition()` از نمونه‌های این دسته هستند.

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

NT: `sample()` – `union()` – `filter()` – `map()`
WT: `aggregate()` – `aggregateByKey()` – `groupByKey()`
Action: `countByKey()` – `count()` – `collect()` – `reduce()`

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.
- برای استفاده از **pyspark** ابتدا باید مسیر فایل نصبی **spark** را از وبسایت **apache spark** پیدا کرده و با استفاده از دستورات زیر آن را از حالت فشرده خارج و با دستور **pip** نصب نماییم. بعد از تعریف مسیر **SPARK_HOME** می‌توانیم از دستورات مرتبط با **pyspark** استفاده نماییم.

```
!wget https://d1cdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
!tar -xvzf spark-3.2.1-bin-hadoop3.2.tgz
!pip install findspark
import os
os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"
import findspark
findspark.init()

--2022-03-02 13:39:20-- https://d1cdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
Resolving d1cdn.apache.org (d1cdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to d1cdn.apache.org (d1cdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 300971569 (287M) [application/x-gzip]
Saving to: 'spark-3.2.1-bin-hadoop3.2.tgz'
```

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

در این قسمت یک لیست ۵۰ تایی به نام bookList از نام کتاب‌ها ساخته شده است.

```
[5] bookList = ['In Search of Lost Time', 'Nineteen Eighty Four', 'The Lord of the Rings', 'Pride and Prejudice', 'The Grapes of Wrath',
               'To Kill a Mockingbird', 'Jane Eyre', 'Wuthering Heights', 'A Passage to India', 'Lord of the Flies',
               'Hamlet', 'A Bend in the River', 'The Great Gatsby', 'The Catcher in the Rye', 'The Bell Jar',
               'Brave New World', 'The Diary of a Young Girl', 'Don Quixote', 'The Bible', 'The Canterbury Tales',
               'The Quiet American', 'Birdsong', 'Money', 'Harry Potter and the Deathly Hallows', 'Harry Potter and the Order of the Phoenix',
               'Harry Potter And The Prisoner Of Azkaban', 'Harry Potter and the Half-Blood Prince', 'Moby Dick', 'The Wind in the Willows', 'His Dark Materials',
               'Anna Karenina', 'Alice Adventures in Wonderland', 'Rebecca', 'On the Road', 'Heart of Darkness',
               'The Way We Live Now', 'The Stranger', 'The Color Purple', 'Life of Pi', 'Frankenstein',
               'War of the Worlds', 'Stories of Ernest Hemingway', 'Gulliver Travels', 'A Christmas Carol', 'Robinson Crusoe',
               'Catch-22', 'The Count of Monte Cristo', 'Memoirs of a Geisha', 'The Divine Comedy', 'The Picture of Dorian Gray']
```

- لیست خود را به RDD تبدیل کنید.

ابتدا یک spark session می‌سازیم و با استفاده از متد parallelize() آن را به RDD تبدیل می‌کنیم.

```
[9] from pyspark.sql import SparkSession
     spark = SparkSession.builder.appName('App1').getOrCreate()
     rdd = spark.sparkContext.parallelize(bookList)
```

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

با استفاده از دستور filter و با کمک تابع یک خطی lambda عنصر ۲۰ام RDD را به دست آورده و با collect() آن را برمی‌گردانیم. در اینجا چون درایه‌ها از صفر شروع می‌شوند عنصر ۲۰ام همان bookList[۱۹] می‌باشد.





```
rdd.filter(lambda x: bookList[19] in x).collect()
```

```
['The Canterbury Tales']
```

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

با استفاده از دستور **map** و تابع یک خطی **lambda** یکی یکی درایه‌ها را با متد **upper()** به صورت حروف بزرگ درآورده و با **collect()** آن‌ها را برمی‌گردانیم.

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

```
 ['IN SEARCH OF LOST TIME',  
'NINETEEN EIGHTY FOUR',  
'THE LORD OF THE RINGS',  
'PRIDE AND PREJUDICE',  
'THE GRAPES OF WRATH',  
'TO KILL A MOCKINGBIRD',  
'JANE EYRE',  
'WUTHERING HEIGHTS',  
'A PASSAGE TO INDIA',  
'LORD OF THE FLIES',  
'HAMLET',  
'A BEND IN THE RIVER',  
'THE GREAT GATSBY',  
'THE CATCHER IN THE RYE',  
'THE BELL JAR',  
'BRAVE NEW WORLD',  
'THE DIARY OF A YOUNG GIRL',  
'DON QUIXOTE',  
'THE BIBLE',  
'THE CANTERBURY TALES',  
'THE QUIET AMERICAN',  
'BIRDSONG',  
'MONEY',  
'HARRY POTTER AND THE DEATHLY HALLOWS',  
'HARRY POTTER AND THE ORDER OF THE PHOENIX',  
'HARRY POTTER AND THE PRISONER OF AZKABAN',  
'HARRY POTTER AND THE HALF-BLOOD PRINCE',  
'MOBY DICK',  
'THE WIND IN THE WILLOWS',  
'HIS DARK MATERIALS',  
'ANNA KARENINA',  
'ALICE ADVENTURES IN WONDERLAND',  
'REBECCA',  
'ON THE ROAD',  
'HEART OF DARKNESS',  
'THE WAY WE LIVE NOW',  
'THE STRANGER',  
'THE COLOR PURPLE',  
'LIFE OF PI',  
'FRANKENSTEIN',  
'WAR OF THE WORLDS',  
'STORIES OF ERNEST HEMINGWAY',  
'GULLIVER TRAVELS',  
'A CHRISTMAS CAROL',  
'ROBINSON CRUSOE',  
'CATCH-22',  
'THE COUNT OF MONTE CRISTO',  
'MEMOIRS OF A GEISHA',  
'THE DIVINE COMEDY',  
'THE PICTURE OF DORIAN GRAY']
```

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

با استفاده از دستور `groupBy` تمام درایه ها را براساس حرف اولشان گروه بندی کرده سپس با استفاده از دستور `map`، گروه ها را به صورت لیست های جدا از هم در می آوریم. بعد از آن با دستور `sortBy` می توانیم خروجی را براساس حرف اول مرتب سازی کنیم.

```

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

[('A',
  ['A Passage to India',
   'A Bend in the River',
   'Anna Karenina',
   'Alice Adventures in Wonderland',
   'A Christmas Carol']),
 ('B', ['Brave New World', 'Birdsong']),
 ('C', ['Catch-22']),
 ('D', ['Don Quixote']),
 ('F', ['Frankenstein']),
 ('G', ['Gulliver Travels']),
 ('H',
  ['Hamlet',
   'Harry Potter and the Deathly Hallows',
   'Harry Potter and the Order of the Phoenix',
   'Harry Potter And The Prisoner Of Azkaban',
   'Harry Potter and the Half-Blood Prince',
   'His Dark Materials',
   'Heart of Darkness']),
 ('I', ['In Search of Lost Time']),
 ('J', ['Jane Eyre']),
 ('L', ['Lord of the Flies', 'Life of Pi']),
 ('M', ['Money', 'Moby Dick', 'Memoirs of a Geisha']),
 ('N', ['Nineteen Eighty Four']),
 ('O', ['On the Road']),
 ('P', ['Pride and Prejudice']),
 ('R', ['Rebecca', 'Robinson Crusoe']),
 ('S', ['Stories of Ernest Hemingway']),
 ('T',
  ['The Lord of the Rings',
   'The Grapes of Wrath',
   'To Kill a Mockingbird',
   'The Great Gatsby',
   'The Catcher in the Rye',
   'The Bell Jar',
   'The Diary of a Young Girl',
   'The Bible',
   'The Canterbury Tales',
   'The Quiet American',
   'The Wind in the Willows',
   'The Way We Live Now',
   'The Stranger',
   'The Color Purple',
   'The Count of Monte Cristo',
   'The Divine Comedy',
   'The Picture of Dorian Gray']),
 ('W', ['Wuthering Heights', 'War of the Worlds'])]

```

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

در این قسمت با استفاده از کتابخانه‌ی **sparkcontext** فایل متن **Input** خوانده شده و با دستورات **map**، **flatMap** و **reduceByKey** خروجی مورد نظر تولید می‌شود. با استفاده از تابع **list** خروجی را در یک لیست قرار می‌دهیم.

```
from pyspark import SparkContext
sc = spark.sparkContext
RddText = sc.textFile("Input")
wordCounts = RddText.flatMap(lambda line: line.split()).map(lambda word:(word, 1)).reduceByKey(lambda a,b:a+b)
list(wordCounts.collect())
```

```
[('Amy', 12),
 ('normally', 1),
 ('Monday', 1),
 ('mornings', 1),
 ('but', 3),
 ('this', 2),
 ('year', 1),
 ('was', 19),
 ('different.', 1),
 ('Kamal', 11),
 ('in', 4),
 ('her', 7),
 ('class', 1),
 ('liked', 1),
 ('She', 4),
 ('waiting', 1),
 ('classroom', 1),
 ('when', 2),
 ('Tara', 14),
 ('Amy!', 1),
 ('sent', 3),
 ('forgot', 2),
 ('inhaler.', 1),
 ('turn', 1),
 ('phone', 3),
 ('on?', 1),
 ('didn't', 10),
 ('like', 7),
 ('never', 2),
 ('messages', 1),
 ('Facebook', 1),
 ('too.', 2),
 ('ask', 1),
 ('best', 1),
 ('know', 2),
 ('everything', 1),
 ('happening', 1),
 ('life.', 1),
 ('"I', 6),
 ('think', 2),
 .
 .
 .
 .
```

- چه تفاوتی بین Action های take و collect وجود دارد؟

دستور take(n) ، آرایه‌ای n تایی از عناصر موجود را برمی‌گرداند در صورتی که collect() همه عناصر موجود را برمی‌گرداند.

- در صورتی که بتوانید توالی انجام هریک از عملیات‌ها در اسپارک که برای هر دستور انجام می‌دهد را برای هریک از دستورات بالا نمایش دهید و با توجه به مفاهیم سوالات قبل آن را تصویر سازی کنید، نمره اضافه‌ای دریافت خواهید کرد. (به کمک ngrok و UI Spark)