

# quiz6

---

CS 119

4/11/2022

Neal Zhao

## Moving Averages

---

1

```
1 from pyspark import SparkContext
2 from pyspark.streaming import StreamingContext
3 from pyspark import StorageLevel
4 import pandas as pd
5 import numpy as np
6 sc.stop()
7 def launch(n):
8     ssc.start()
9     ssc.awaitTermination(n)
10
11 sc = SparkContext('local[2]', 'NetworkWordCount')
12 ssc = StreamingContext(sc,1)
13 lines = ssc.socketTextStream('localhost', 9999,
    StorageLevel.MEMORY_AND_DISK)
```

2

```
1 # q2
2 from pyspark import SparkContext
3 from pyspark.streaming import StreamingContext
4 from pyspark import StorageLevel
5 import pandas as pd
6 import numpy as np
7
8 sc.stop()
9
10 def launch(n):
11     ssc.start()
12     ssc.awaitTermination(n)
13
14 def calculate_sum(rdd):
15     l = rdd.collect()
16     if len(l) == 80:
17         total10 = 0
18         for i in range(60, len(l)):
19             if i%2 == 1:
20                 total10 += float(l[i])
21
22         total40 = 0
23         for i in range(len(l)):
24             if i%2 == 1:
```

```

25         total40 += float(l[i])
26         return sc.parallelize([(total40, total10)])
27
28 def dj_count(rdd):
29     return sc.parallelize([len(rdd.collect())])
30
31
32 sc = SparkContext('local[2]', 'NetworkWordCount')
33 ssc = StreamingContext(sc,1)
34 lines = ssc.socketTextStream('localhost', 9999,
StorageLevel.MEMORY_AND_DISK)
35 words = lines.window(40,1).flatMap(lambda line:line.split(' '))
36 dj30sum = words.transform(calculate_sum)
37 dj30count = words.transform(dj_count)
38
39
40 launch(10)

```

### 3

```

1 def window_both(rdd):
2     l = rdd.collect()
3     if len(l) == 80:
4         total10 = 0
5         for i in range(60, len(l)):
6             if i%2 == 1:
7                 total10 += float(l[i])
8         total40 = 0
9         for i in range(len(l)):
10             if i%2 == 1:
11                 total40 += float(l[i])
12         return sc.parallelize([total40/40, total10/10])
13
14 words = lines.window(40,1).flatMap(lambda line:line.split(' '))
15 maBoth = words.transform(window_both)
16 maBoth.foreachRDD(lambda x: print('40 MA: ', x.collect()[0], '10 MA: ',
x.collect()[1]))
17
18 launch(10)

```

### 4

```

1 from pyspark import SparkContext
2 from pyspark.streaming import StreamingContext
3 from pyspark import StorageLevel
4 import pandas as pd
5 import numpy as np
6
7 sc.stop()
8
9
10 global records
11 records = [0,0,0]
12

```

```

13 def window_both(rdd):
14     l = rdd.collect()
15     if len(l) == 80:
16         total10 = 0
17         for i in range(60, len(l)):
18             if i%2 == 1:
19                 total10 += float(l[i])
20         total40 = 0
21         for i in range(len(l)):
22             if i%2 == 1:
23                 total40 += float(l[i])
24         return sc.parallelize([l[78], total40/40, total10/10])
25
26
27 def store_record_and_check(rdd):
28     global records
29     if rdd is None:
30         return
31     l = rdd.collect()
32     last_record = records
33     current_record = l.copy()
34     records = current_record
35     if last_record[1] > last_record[2]: #40 above 10
36         if current_record[1] < current_record[2]:# 40 below 10, buy
37             return sc.parallelize([current_record[0], 'BUY DJ30'])
38     elif last_record[1] < last_record[2]: #40 below 10
39         if current_record[1] > current_record[2]:# 40 above 10, sell
40             return sc.parallelize([current_record[0], 'sell DJ30'])
41
42
43
44
45
46 def launch(n):
47     ssc.start()
48     ssc.awaitTermination(n)
49
50
51 sc = SparkContext('local[2]', 'NetworkWordCount')
52 ssc = StreamingContext(sc,1)
53 lines = ssc.socketTextStream('localhost', 9999,
54                               StorageLevel.MEMORY_AND_DISK)
55 words = lines.window(40,1).flatMap(lambda line:line.split(' '))
56 maBoth = words.transform(window_both)
57 result = maBoth.transform(store_record_and_check)
58 result.foreachRDD(lambda x: print(x.collect()[0], x.collect()[1]))
59 launch(10)

```

printed result:

(The stream is not ended)

1	3/8/90 BUY DJ30
---	-----------------

2	4/26/90	sell	DJ30
3	5/14/90	BUY	DJ30
4	8/2/90	sell	DJ30
5	12/31/90	BUY	DJ30
6	1/10/91	sell	DJ30
7	1/25/91	BUY	DJ30
8	3/29/91	sell	DJ30
9	4/5/91	BUY	DJ30
10	4/10/91	sell	DJ30
11	4/16/91	BUY	DJ30
12	5/16/91	sell	DJ30
13	5/31/91	BUY	DJ30
14	6/28/91	sell	DJ30
15	7/15/91	BUY	DJ30
16	7/17/91	sell	DJ30
17	7/19/91	BUY	DJ30
18	8/22/91	sell	DJ30
19	8/23/91	BUY	DJ30
20	9/23/91	sell	DJ30
21	9/24/91	BUY	DJ30
22	10/7/91	sell	DJ30
23	10/21/91	BUY	DJ30
24	11/19/91	sell	DJ30
25	12/27/91	BUY	DJ30
26	3/11/92	sell	DJ30
27	3/25/92	BUY	DJ30
28	4/3/92	sell	DJ30
29	4/15/92	BUY	DJ30
30	6/17/92	sell	DJ30

```
31 8/3/92 BUY DJ30
32 8/19/92 sell DJ30
33 9/21/92 BUY DJ30
34 9/28/92 sell DJ30
35 11/5/92 BUY DJ30
36 1/14/93 sell DJ30
37 2/3/93 BUY DJ30
38 4/14/93 sell DJ30
39 4/19/93 BUY DJ30
40 4/28/93 sell DJ30
41 5/11/93 BUY DJ30
```

# Bloom Filter

## 1

```
1 data = open('headline_words.txt').read().split()
2 n = len(data)*8
3 import numpy as np
4 filter_bit = np.zeros(n, int)
5 for word in data:
6     position = hash(word) % n
7     filter_bit[position] = 1
8 np.savetxt('arr.txt', filter_bit, fmt="%d", newline = ' ')
```

## 2

```
1 from pyspark import SparkContext
2 from pyspark.streaming import StreamingContext
3 from pyspark import StorageLevel
4 import pandas as pd
5 import numpy as np
6 import re
7
8 global word_l
9 word_l = np.array(open('bloom.txt').read().split(), int)
10 # print(word_l)
11
12 # takes in a RDD containing a line,
13 # print it if it has unfamiliar words
14 def foo(rdd):
15     global word_l # the bit array of familiar words
16     line = rdd.collect()
```

```

17     if len(line) == 0:
18         return
19     text = line[0] #get the string
20     text = text.lower()
21     text = re.sub("'", "", text)
22     text = re.sub(r'[0-9\,$]+', '', text)
23     text = re.sub("(\d|\\w)+", " ", text)
24     text = ' '.join([word for word in text.split() if len(word) > 3])
25     words = text.split(' ')
26     for word in words:
27         pos = hash(word) % len(word_1)
28         if word_1[pos] == 1:
29             print(line[0])
30             break
31     return
32
33
34
35 def launch(n):
36     ssc.start()
37     ssc.awaitTermination(n)
38
39
40 sc.stop()
41 sc = SparkContext('local[2]', 'NetworkWordCount')
42 ssc = StreamingContext(sc, 1)
43 lines = ssc.socketTextStream('localhost', 9999,
44                               StorageLevel.MEMORY_AND_DISK)
45 lines.foreachRDD(foo)
46 launch(10)

```

Link:

[https://tufts.zoom.us/rec/share/sZcsDgB2wPbVN-glgywexBPO8\\_o73e3Q-o2cFUbkmr8RReLNfXha1m\\_Uj2IMDuH.3wCDwftW4TpRQozj?startTime=1649819763000](https://tufts.zoom.us/rec/share/sZcsDgB2wPbVN-glgywexBPO8_o73e3Q-o2cFUbkmr8RReLNfXha1m_Uj2IMDuH.3wCDwftW4TpRQozj?startTime=1649819763000)