ex46

August 12, 2022

[1]:
```
# Solution Ex. 46
```

[2]:
```
import sys
```

[3]:
```
inputPath = "data/Ex46/data/readings.txt" # "/data/students/bigdata-01QYD/
 ↪ex_data/Ex46/data/readings.txt"
outputPath = "res_out_Ex46v2/"
```

[4]:
```
# Read the content of the readings
readingsRDD = sc.textFile(inputPath)
```

[9]:
```
readingsRDD.collect()
```

[9]:
```
['1451606400,12.1',
 '1451606460,12.2',
 '1451606520,13.5',
 '1451606580,14.0',
 '1451606640,14.0',
 '1451606700,15.5',
 '1451606760,15.0']
```

[5]:
```
# Generate the elements of each window.
# Each reading with start time t belongs to 3 windows with a window size equal␣
 ↪to 3:
# - The one starting at time t-120s
# - The one starting at time t-60s
# - The one starting at time t

def windowElementsFunc(reading):
    fields = reading.split(",")

    # Time stamp of this reading
    t = int(fields[0])
    # Temperature
    temperature = float(fields[1])

    # The current reading, associated with time stamp t,
```

```
    # is part of the windows starting at time t, t-60s, t-120s

    # pairs is a list containing three pairs (window start timestamp, current reading) associated with
    # the three windows containing this reading
    pairs = []

    # Window starting at time t
    # This reading is the first element of the window starting at time t
    pairs.append((t, reading))

    # Window starting at time t-60
    # This reading is the second element of that window starting at time t-60
    pairs.append((t-60, reading))

    # Window starting at time t-120
    # This reading is the third element of that window starting at time t-120
    pairs.append((t-120, reading))

    return pairs
```

[6]:
```
windowsElementsRDD = readingsRDD.flatMap(windowElementsFunc)
```

[7]:
```
# Use groupByKey to generate one sequence for each time stamp
timestampsWindowsRDD = windowsElementsRDD.groupByKey()
```

[8]:
```
timestampsWindowsRDD.mapValues(lambda v: list(v)).collect()
```

[8]:
```
[(1451606400, ['1451606400,12.1', '1451606460,12.2', '1451606520,13.5']),
 (1451606340, ['1451606400,12.1', '1451606460,12.2']),
 (1451606280, ['1451606400,12.1']),
 (1451606460, ['1451606460,12.2', '1451606520,13.5', '1451606580,14.0']),
 (1451606520, ['1451606520,13.5', '1451606580,14.0', '1451606640,14.0']),
 (1451606580, ['1451606580,14.0', '1451606640,14.0', '1451606700,15.5']),
 (1451606640, ['1451606640,14.0', '1451606700,15.5', '1451606760,15.0']),
 (1451606700, ['1451606700,15.5', '1451606760,15.0']),
 (1451606760, ['1451606760,15.0'])]
```

[10]:
```
# This function is used in the next transformation to select the windows with an incrasing temperature trend
def increasingTrendFunc(pairInitialTimestampWindow):

    # The key of the input pair is the intial timestamp of the current window
    minTimestamp = pairInitialTimestampWindow[0]

    # Store the (at most) 3 elements of the window in a dictionary
    # containing enties time stamp -> temperature
```

```
    timestampTemp = {}

    # pairInitialTimestampWindow[1] contains the elements of the current window
    window = pairInitialTimestampWindow[1]


    for timestampTemperature in window:
        fields = timestampTemperature.split(",")
        t = int(fields[0])
        temperature = float(fields[1])

        timestampTemp[t] = temperature


    # Check if the list contains three elements.
    # If the number of elements is not equal to 3 the window is incomplete and must be discarded
    if len(timestampTemp) != 3:
        increasing = False
    else:
        # Check is the increasing trend is satisfied
        if timestampTemp[minTimestamp]<timestampTemp[minTimestamp+60] and timestampTemp[minTimestamp+60]<timestampTemp[minTimestamp+120]:
            increasing = True
        else:
            increasing = False

    return increasing
```

[11]:
```
seletedWindowsRDD = timestampsWindowsRDD.filter(increasingTrendFunc)
```

[31]:
```
# The result is in the value part of the returned pairs
```

[12]:
```
seletedWindowsRDD.values().map(lambda window: list(window)).collect()
```

[12]:
```
[['1451606400,12.1', '1451606460,12.2', '1451606520,13.5'],
 ['1451606460,12.2', '1451606520,13.5', '1451606580,14.0']]
```

[19]:
```
# Store the result. Map the iterable associated with each window to a list
```

[20]:
```
seletedWindowsRDD.values().map(lambda window: list(window)).
 ↪saveAsTextFile(outputPath)
```

[ ]:
```

```