데이터과학

5주차 과제 - 2016 Sokulee 데이터 분석

충남대학교 컴퓨터공학과
201402395 이 승 희
제출일자 : 2017 - 04 - 05

실습 보고서

1. 과제 개요

2016년도 소쿠리대회 데이터를 기반으로 대회 기간 동안의 걸음수 Top10을 구한다.

2. 구현 내용

main의 내용이다. 직접 구현한 merging_users_data()로부터 다중의 steps.json file 을 각각 구분하여 users_data에 저장한다.

이후 걸음수 Top10을 위해 각 json 데이터로부터 총 걸음수 값만을 추출하여 users_steps에 저장한다.

```
46      # main
47      users_data = merging_users_data(get_paths())
48      users_steps = get_each_step_datas_in_users(users_data)
```

다음은 직접 구현한 메소드들의 내용이다. get_paths() 함수는 .py 파일과 같은 디렉토리에 있는 sokulee 폴더 내 A01-A098 User 데이터에 접근하기 위한 각 path를 구하기 위함이다. 이 함수로부터 반환된 array는 merging_users_data() 함수 호출 시 매개변수로서 사용된다.

merging_users_data는 각 path에서 ~steps를 포함하는 json 파일만을 load한다. 이 때 get_merges_json 함수를 호출하여 각 사용자 데이터 내 steps.json 파일을 load한 후 list에 저장하여 반환한다. 또한 1부터 98까지 모든 user가 존재하지는 않기 때문에 향후 폴더 내 존재하는 user만 검사하기 위하여 사용자의 numbering을 저장할 user_indices라는 임의의 배열을 선언하고, file directory를 검사하여 값들을 저장한다.

결과적으로 merging_users_data() 함수의 반환값으로서 main 내 users_data에 저장된다

```
9       # get current file directory path
10      def get_paths():
11          paths = []
12          for i in range(1, 99):
13              paths.append(sys.path[0] + "/sokulee/A0" + str(i))
14          return paths
15
16      """
17       1부터 98까지 모든 user가 존재하진 않기 때문에
18       향후 폴더 내 존재하는 number의 user만 체크하기 위해 array 선언
19      """
20      user_indices = []
21      # get each json datas of multiple files
22      def get_merges_json(flist):
23          temp = []
24          for f in flist:
25              i = 0
26              for str in f.split("/"):
27                  if "A0" not in str: i+= 1
28                  else: break
29              if f.split("/")[i] not in user_indices:
30                  user_indices.append(f.split("/")[i])
31              with open(f) as result:
32                  data = json.load(result)
33                  temp.append(data)
34          return temp
35
36      # merge the data of each user's steps into list
37      def merging_users_data(paths):
38          datas = []
39          for path in paths:
40              file_mask = os.path.join(path, '*steps.json')
41              temp = []
42              for values in get_merges_json(glob.glob(file_mask)):
43                  temp.append(values)
44              datas.append(temp)
45          return datas
```

다음 함수는 users_data 중에서도 실습에 사용될 걸음수 만을 추출하기 위한 함수이다. 각 사용자 데이터 중에서도 activities-steps가 존재하지 않는 경우가 있어 이를 예외처리하였다.

```
34      # extract the total value of steps of each users
35      def get_each_step_datas_in_users(users_data):
36          users = []
37          for user in users_data:
38              total_steps = 0
39              for single_steps in user:
40                  if 'activities-steps' in single_steps:
41                      result = json_normalize(single_steps['activities-steps'])
42                      total_steps += int(result['value'])
43              users.append(total_steps)
44          return users
```

위 두 과정을 거치고 나면 다음과 같이 정렬되지 않은 형태의 98개 사용자 데이터가 추출된다.

```
iseunghuiui-MacBook:data science leeseunghee$ python3 sokulee_test.py
[351518, 317873, 391302, 667043, 205815, 688864, 584462, 616758, 0, 266725, 0, 0, 0, 0, 0, 535526, 697
669, 800509, 802587, 676258, 290324, 439406, 0, 759414, 1083320, 1070291, 464660, 421421, 626306, 4777
48, 325301, 728916, 414915, 357887, 503583, 244243, 413353, 775284, 563457, 727438, 409936, 574088, 97
4336, 580578, 835907, 0, 413737, 294072, 325833, 380282, 0, 626213, 289223, 810861, 0, 544578, 127678,
 270693, 343124, 385730, 489414, 816567, 414915, 0, 464660, 0, 0, 584932, 0, 0, 406062, 458095, 289141
, 0, 0, 0, 0, 0, 0, 413353, 423731, 0, 802587, 443310, 0, 0, 0, 0, 0, 0, 0, 559912, 499193, 665433, 0,
 502283, 276376, 295227]
```

이를 내림차순으로 정렬하고, top 10 데이터만을 사용하기 위해 다음 과정을 거치게 된다. 먼저 steps가 0일 경우가 존재하지 않는 사용자이기 때문에 이를 제외한 사용자들을 dictionary로 재정의한다. 이 결과로서 출력된 extracted_data는 아래와 같다.

```
iseunghuiui-MacBook:data science leeseunghee$ python3 sokulee_test.py
{'A01': 351518, 'A02': 317873, 'A03': 391302, 'A04': 667043, 'A05': 205815, 'A06': 688864, 'A07': 584462, 'A08': 616758, 'A010': 26
6725, 'A016': 535526, 'A017': 697669, 'A018': 800509, 'A019': 802587, 'A020': 676258, 'A021': 290324, 'A022': 439406, 'A024': 75941
4, 'A025': 1083320, 'A026': 1070291, 'A027': 464660, 'A028': 421421, 'A029': 626306, 'A030': 477748, 'A031': 325301, 'A032': 728916
, 'A033': 414915, 'A034': 357887, 'A035': 503583, 'A036': 244243, 'A037': 413353, 'A038': 775284, 'A039': 563457, 'A040': 727438, '
A041': 409936, 'A042': 574088, 'A043': 974336, 'A044': 580578, 'A045': 835907, 'A047': 413737, 'A048': 294072, 'A049': 325833, 'A05
0': 380282, 'A052': 626213, 'A053': 289223, 'A054': 810861, 'A056': 544578, 'A057': 127678, 'A058': 270693, 'A059': 343124, 'A060':
 385730, 'A061': 489414, 'A062': 816567, 'A063': 414915, 'A065': 464660, 'A068': 584932, 'A071': 406062, 'A072': 458095, 'A073': 28
9141, 'A080': 413353, 'A081': 423731, 'A083': 802587, 'A084': 443310, 'A092': 559912, 'A093': 499193, 'A094': 665433, 'A096': 50228
3, 'A097': 276376, 'A098': 295227}
iseunghuiui-MacBook:data science leeseunghee$
```

다시 이 데이터를 내림차순으로 정렬하고, 사용하게될 top 10 데이터만 다시 저장한다.

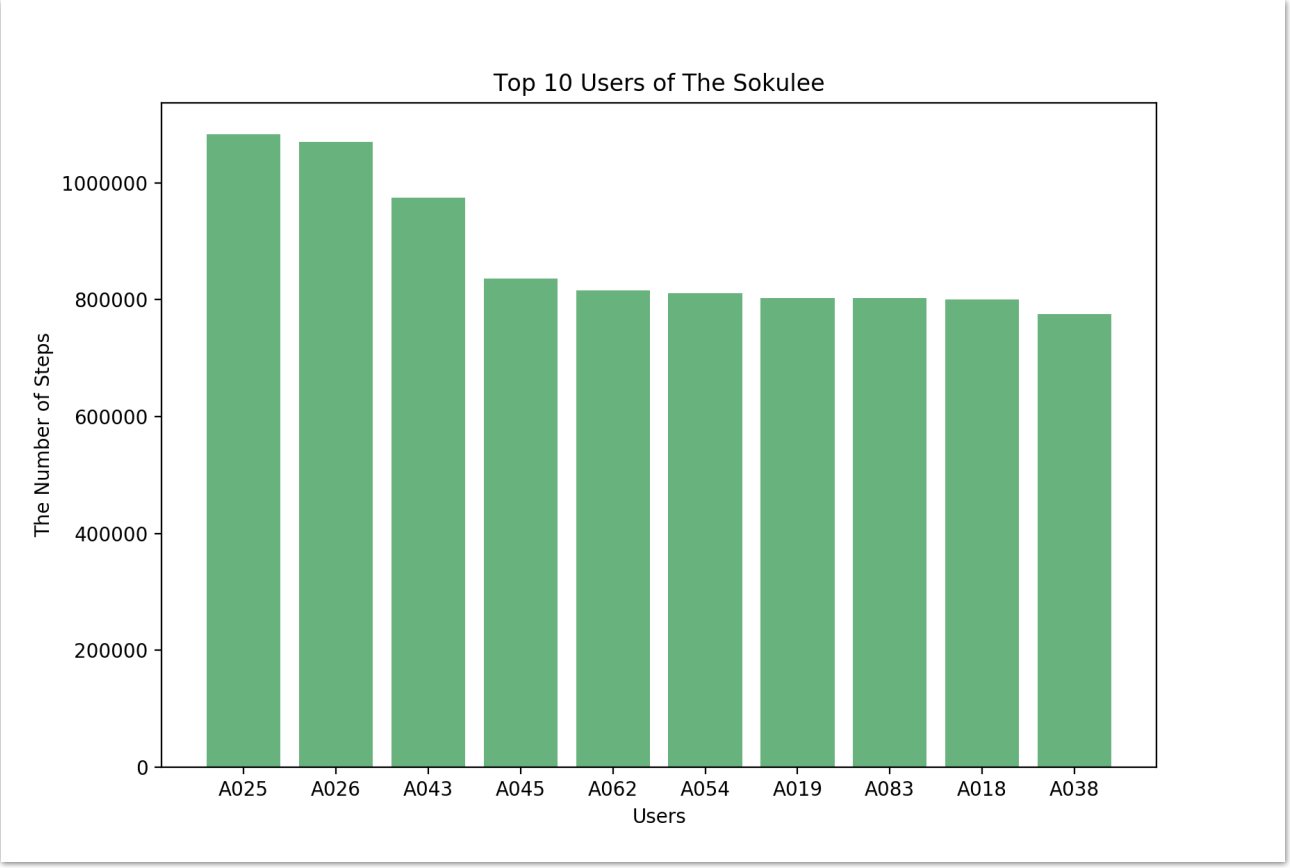
```
60      # extract the users who has usable data
61      extracted_data = {}
62      index = 0
63      for step in users_steps:
64          if step is not 0:
65              extracted_data[user_indices[index]] = step
66              index += 1
67
68      # sorting the sorted_result by descending
69      from operator import itemgetter
70      sorted_result = sorted(extracted_data.items(), key=itemgetter(1), reverse=True)
71      sorted_result = sorted_result[:10]
```

위 과정을 모두 거친 뒤 최종적으로 그래프를 그릴 수 있다. sorted_result가 itemgetter의 결과로서 'tuple' object의 list이기 때문에 이를 x_axis, y_axis로 구분하여 정의한 후 각각 x축, y축으로 plot하였다.

```
73      # draw the graph
74      import matplotlib.pyplot as plt
75
76      x_axis = []
77      y_axis = []
78
79      for temp_tuple in sorted_result:
80          x_axis.append(temp_tuple[0])
81          y_axis.append(temp_tuple[1])
82
83      plt.bar(range(len(x_axis)), y_axis, color='#68b37d')
84      plt.xticks(range(len(x_axis)), x_axis)
85      plt.xlabel('Users')
86      plt.ylabel('The Number of Steps')
87      plt.title('Top 10 Users of The Sokulee')
88      plt.show()
```

3. 과제 결과

실행 결과, 다음과 같은 결과를 도출할 수 있었다. 가공된 데이터를 사용하는 것은 어렵지 않았지만, 가공하기까지 각 사용자, 일자 별로 데이터가 많은데다가 검사, 비교하는 과정도 복잡했고, 무엇보다 Python api들이 너무 많은데다 익숙치 않아서 데이터를 처리하는게 가장 어려웠다.



과제 개요

실습에서 다뤘던 2016 소쿠리 데이터 분석 대회의 데이터를 분석하여, 해결하고자 하는 문제를 정의하고, 직접 결과를 도출해낸다.

내가 정의한 문제는 첫째로 대회기간동안 날짜에 따른 사용자들의 수면 시간의 평균량과, 수면시간 중 awake와 sleep을 다시 구분하여 이들의 관계도 파악해 보는 것이었다. 또한 요일별로 수면량의 평균을 정의하여 이들의 관계는 어떻게 되는지 확인하고자 한다.

과제 내용

다음은 main의 내용이다. 앞서 실습에서 사용한 메소드들을 이용하되, 이전에는 'steps'들을 포함한 json 파일들을 사용했다면, 여기서는 sleep 데이터를 이용할 것이기 때문에 title을 별도의 parameter로 정의하여 'sleep'을 포함한 json 파일들을 loading하였다.

그런데 데이터들을 처리하던 중, 일부 날짜에서는 sleep 데이터가 기록되지 않은 것을 확인하였고, 결국 사용자마다 기록된 날짜가 모두 달랐다. 하지만 결과물로서는 대회 내 기간이 모두 출력되어야 하기 때문에 기록된 모든 date들을 추출하여 logged_dates를 정의하였다.

```
75      """""""""""""""""""""""""""
76       ******* main *******
77      """""""""""""""""""""""""""
78      users_data = merging_users_data(get_paths(), 'sleep')
79      users_sleeps = get_each_sleeps_data_in_users(users_data)
80
81      """
82       사용자마다 기록되지 않은 date가 존재하기 때문에
83       데이터가 기록된 모든 date를 추출하기 위해 logged_dates를 정의한다
84      """
85      logged_dates = []
86      for user in users_sleeps:
87          for key in users_sleeps[user]:
88              if key not in logged_dates:
89                  logged_dates.append(key)
```

다음은 위 main에서 호출된 함수 get_each_sleeps_data_in_users()의 내용이다.

앞서 언급했듯, 필요한 데이터는 sleep 필드의 'dateOfSleep', summary 필드의 'minutesAsleep', 'minutesAwake', 'totalMinutesAsleep' 데이터들이다. 각 json 파일들의 내용을 검사하여 이들을 저장하는데, sleep 필드의 경우 하루 중 sleep log가 여러 개일 수 있기 때문에 dateOfSleep이 중복되는 경우가 발생하여 이를 예외처리 하였다.

```
45      """
46       각 user의 sleep data를 추출
47
48       sleep: 기록된 수면 로그들의 집합 (없을수도, 한 개일수도, 여러개일수도)
49       dateOfSleep: 수면 로그가 기록된 날짜
50       minutesAsleep: 실제 수면을 취한 시간(분)
51       minutesAwake: 수면 중 awake한 시간(분)
52       totalMinutesAsleep: 총 수면을 취한 시간(분)
53      """
54      def get_each_sleeps_data_in_users(users_data):
55          users = {}
56          index = 0
57          for user in users_data:
58              users_data = {}
59              for single_date in user:
60                  if 'summary' in single_date and 'sleep' in single_date:
61                      summary_result = json_normalize(single_date['summary'])
62                      if len(single_date['sleep']) > 1:
63                          sleep_logs = json_normalize(single_date['sleep'][0])
64                      elif len(single_date['sleep']) == 1:
65                          sleep_logs = json_normalize(single_date['sleep'])
66                      date = sleep_logs['dateOfSleep'].iloc[0]
67                      users_data[date] = \
68                          {'minutesAsleep': int(summary_result['totalMinutesAsleep']),
69                          'minutesAwake':int(summary_result['totalTimeInBed']) - int(summary_result['totalMinutesAsleep'])}
70              if len(users_data) > 0:
71                  users[user_name_indices[index]] = users_data
72                  index += 1
73          return users
74
```

위 과정을 거치면 결과적으로 사용자들의 각 데이터에 대한 내용이 dictionary로 저장된다.

```
93      # 사용자의 time_for_awake 데이터와 time_for_sleep 데이터 정의
94      time_for_awake = []
95      time_for_sleep = []
96      for date in logged_dates:
97          total_awake = 0
98          total_sleep = 0
99          for user in users_sleeps:
100             if date in users_sleeps[user].keys():
101                 total_sleep += users_sleeps[user][date]['minutesAsleep']
102                 total_awake += users_sleeps[user][date]['minutesAwake']
103             else:
104                 total_sleep += 0
105                 total_awake += 0
106         time_for_awake.append(total_awake / len(users_sleeps))
107         time_for_sleep.append(total_sleep / len(users_sleeps))
```

첫번째로 확인할 내용인 사용자들의 time_for_awake와 time_for_sleep의 내용이다. 앞서 언급한 get_each_sleeps_data_in_users() 함수의 반환형으로서 users_sleeps를 다루는 내용이다. 결과적으로는 평균량만 사용할 것이기 때문에 사용자들의 수면 데이터들의 평균값을 저장하였다.

```
109     # 총 수면 데이터를 요일별로 재정의
110     import pandas as pd
111
112     sleep_for_weeks = []
113     weeks = pd.DatetimeIndex(logged_dates).weekday.tolist()
114
115     for i in range(0, 7):
116         total = 0
117         for value in weeks:
118             if value == i:
119                 total += (time_for_awake[weeks.index(value)] + time_for_sleep[weeks.index(value)])
120         sleep_for_weeks.append(total)
```

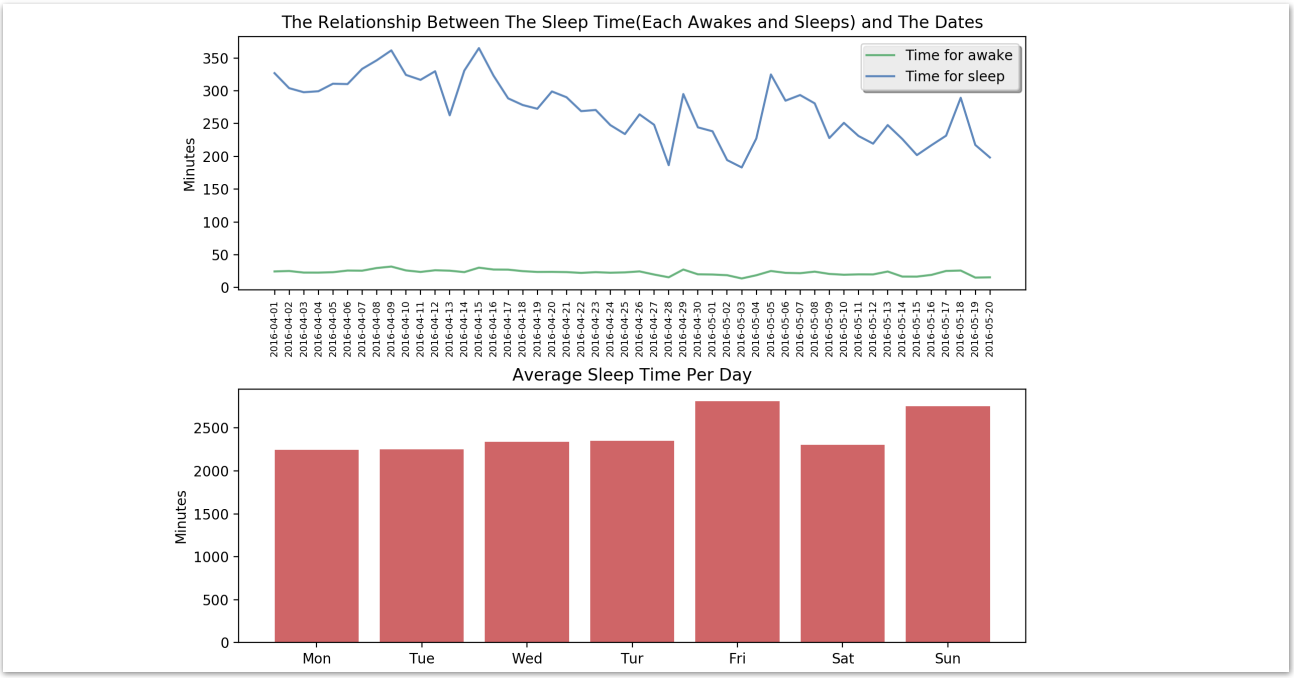
두번째로 확인할 내용은 요일별 수면량의 관계였다. 이전 타슈 데이터를 다룰때 참고했었던 weekday() 함수를 사용하여 logged_dates들을 weeks로 다시 정의하였고, 각 요일별 수면량(결과적으로 awake와 sleep을 합한 값) 들을 total로 계산하였다.

위 과정들을 모두 거치고 나서 그래프들을 그리는 부분이다. 앞서 logged_dates를 정의할때 이전 logged_dates list에 해당 date가 없을 경우 단순히 append하는 형식이었기 때문에 logged_dates는 정렬되지 않은 형태이다. 이를 날짜순으로 정렬하기 위해 date object로 변환 후 정렬, 이후 다시 str형태로 저장하였다. 다른 부분들은 단순히 그래프를 그리는 부분이다.

```
122     # draw the two graphs
123     import matplotlib.pyplot as plt
124     import numpy as np
125     import datetime
126
127     fig = plt.figure(1)
128     width = 0.8
129
130     # logged date는 날짜순이 아닌 배열이므로 이를 날짜순으로 재정렬 (실질적인 데이터는 날짜순으로 기록되어있을 것)
131     x_axis = [datetime.datetime.strptime(date, "%Y-%m-%d") for date in logged_dates]
132     x_axis.sort()
133     x_axis = [datetime.datetime.strftime(date, "%Y-%m-%d") for date in x_axis]
134
135     # the first graph
136     graph = fig.add_subplot(211)
137     awake = graph.plot(range(len(x_axis)), time_for_awake, color='#68b37d', label="Time for awake")
138     sleep = graph.plot(range(len(x_axis)), time_for_sleep, color='#5f88bc', label="Time for sleep")
139     graph.set_xticks(np.arange(len(x_axis)))
140     graph.set_xticklabels(x_axis, rotation=90, fontsize=7)
141     graph.legend(loc='best', shadow=True, fancybox=True, numpoints=1)
142     plt.ylabel('Minutes')
143     plt.title('The Relationship Between The Sleep Time(Each Awakes and Sleeps) and The Dates')
144
145     # the second graph
146     graph = fig.add_subplot(212)
147     week_name = ['Mon', 'Tue', 'Wed', 'Tur', 'Fri', 'Sat', 'Sun']
148     graph.bar(range(0, 7), sleep_for_weeks, width=width, color='#cf6567', align='center')
149     graph.set_xticks(np.arange(len(week_name)))
150     graph.set_xticklabels(week_name)
151     plt.ylabel('Minutes')
152     plt.title('Average Sleep Time Per Day')
153
154     plt.show()
```

결론 및 고찰

앞서 다루고자 하는 데이터들을 정의, 처리한 후 결과물은 다음과 같다.



첫번째인 사용자들의 일자별 awake, sleep 데이터의 평균량 그래프이다. awake 같은 경우 비등비등해서 어떠한 관계를 도출해내기는 힘들지만, sleep의 경우 전체적으로 감소하는 것을 확인할 수 있었다. 하지만 중간에 급격하게 감소하는 부분은 기간상 중간고사라 학생들의 평균 수면시간이 감소하지 않았나하는 개인적인 의견이다. 또한, 데이터를 처리하는 과정에서 사용자들이 대체적으로 시간이 흐르면서 데이터의 빈도수가 적었다. 아무래도 매일같이 데이터를 수집하는 것이 무리가 있기때문이 아닌가 싶다.

위 데이터만으로는 어떠한 관계를 도출해내기는 무리가 있어서, 요일별 수면량을 다시 도출해내보았다. 우선 금요일같은 경우 유흥을 즐기는 학생들이 많기에 수면량이 가장 적지 않았나 싶다.(금요일에서 토요일로 넘어가기 때문에 요일상으로는 토요일에 해당한다.) 또한 주말같은 경우(결과데이터로서는 일요일, 월요일에 해당하게 된다.)는 쉬는 학생들이 많기 때문에 수면량이 대체적으로 많은 것을 확인할 수 있었다.

지금까지 한 과제들 중에서 가장 어려운 난이도에 해당하지 않았나 싶다. 실습 보고서에서도 언급했듯이 데이터를 다루는 과정이 가장 힘들었고, 과제의 난이도도 난이도지만 어떤 문제를 정의해야할지 또한 힘들게 고민했던 것 같다.

결과로서 어떤 명확한 결론을 정의하기는 힘들지만, 시험기간엔 학생들의 수면량이 급격하게 감소하는것, 금요일 같은 경우 유흥을 즐기는 학생들이 많다는 것 정도는 확인할 수 있었다. 본인도 이에 해당하는 것 같아서 결과를 확인했을 때 흥미롭기도 했다.