Week3

19377419 孙启航 文本处理——情绪理解

Code

```
import jieba.analyse as analyse
import numpy as np
import jieba
import time
import re


def jieba_set_config():
    '''
    对jieba库设定参数，添加用户词典，同时设置停用词（停用词引用了上次作业中
stopword_list.txt)
    '''
    print("\nSetting Jieba module config.\n")

    jieba.load_userdict(
        "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/anger.txt")
    jieba.load_userdict(
        "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/disgust.txt")
    jieba.load_userdict(
        "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/fear.txt")
    jieba.load_userdict(
        "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/joy.txt")
    jieba.load_userdict(
        "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/sadness.txt")

    analyse.set_stop_words(
        "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/stopwords_list.txt")


def clear_data(file):
    '''
    进行数据清洗和整理

    去除URL，将正文内容，时间和地点分别提取出，并和微博内容组成一个维度为3的列表，最终组成一个
矩阵
    '''
    with open(file, 'r', encoding='utf-8') as f:
        data = f.readlines()

    clean_data = []
    for i in data:
        vector = [0 for i in range(3)]

        pattern = re.compile(
            r'(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:
[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()
<>]+\)))*\)|[^\s`!()\[\]{};:\'".,<>?«»“”‘’]))')
```

```
        i = re.sub(pattern, '', i)  # 去除微薄的URL等无关信息

        i = i.replace("转发微博", '')

        pattern = re.compile(r'(\[[^]]*)\d(?=[^\]]*\])')

        position = re.findall(pattern, i)

        position = position[-1].replace('[', '')
        # 用正则表达式选出的pattern仍有一点缺陷，所以选出的position多了一个中括号，可以继续
加以改进
        # 在这里就用最基础的方法多进行一次处理
        position = list(map(float, position.split(",")))

        vector[2] = position
        # print(i)
        i = re.sub(pattern, '', i)

        i = i.rstrip()
        i = i[:-1]
        # 去掉右侧的空格，然后再去掉最后的方括号

        timestring = i[-31:]
        # timestring = timestring.replace("+0800",'')
        timestring = timestring.split("\r")[0]
        timestring = timestring.split("\t")[0]

        vector[1] = time.strptime(timestring, "%a %b %d %H:%M:%S +0800 %Y")

        vector[0] = i.replace(timestring, '')
        clean_data.append(vector)

    pattern = re.compile(r"(回复)?(//)?\s*@\S*?\s*(:| |$)")

    timeseries = []
    for i in clean_data:
        i[0] = re.sub(pattern, '', i[0])
        i[0] = analyse.extract_tags(i[0])
        # print(i)

    return clean_data


def analysis1(clean_data):
    '''
    已经完成清洗和整理的数据进行分析

    分别与五个字典进行比较，返回一个矩阵

    矩阵的五列分别对应anger,disgust,fear,joy,sadness
    '''
    def convert(lis):
        '''
        对五个字典进行处理，返回一个中文词的列表
        '''
        lis = ''.join(lis).strip("\n").splitlines()
        return lis
    with
open("C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/anger.txt", 'r',
encoding='utf-8') as anger:
        anger_list = anger.readlines()
        anger_list = convert(anger_list)

    with
open("C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/disgust.txt",
'r', encoding='utf-8') as anger:
        disgust_list = anger.readlines()
        disgust_list = convert(disgust_list)

    with
open("C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/fear.txt", 'r',
encoding='utf-8') as anger:
        fear_list = anger.readlines()
        fear_list = convert(fear_list)

    with
open("C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/joy.txt", 'r',
encoding='utf-8') as anger:
        joy_list = anger.readlines()
        joy_list = convert(joy_list)

    with
open("C:/Users/DELL/Desktop/Code/BUAA_21/Week3/emotion_lexicon/sadness.txt",
'r', encoding='utf-8') as anger:
        sadness_list = anger.readlines()
        sadness_list = convert(sadness_list)

    def judge(matrix):
        nonlocal anger_list, disgust_list, fear_list, joy_list, sadness_list,
clean_data

        for comment in clean_data:
            vector = [0 for i in range(5)]
            info = []
            for word in comment[0]:
                if word in anger_list:
                    vector[0] += 1
                elif word in disgust_list:
                    vector[1] += 1
                elif word in fear_list:
                    vector[2] += 1
                elif word in joy_list:
                    vector[3] += 1
                elif word in sadness_list:
                    vector[4] += 1
            sum_num = sum(vector)
            if sum_num != 0:
                # 加入一步判断，去掉无情绪的向量
                for i in range(5):
                    vector[i] /= float(sum_num)
                info.append(vector)
                # 筛选出符合实际时间范围的数据
                info.extend(comment[1:])
                matrix.append(info)
                # print(info)
            else:
                pass

        return matrix

    return judge


def emotion_trend(Matrix, mood, mode):
    '''
    对情绪变化的趋势进行时间分析

    而通过观察时间的分布范围发现，总体的时间范围是2013-10-11 至 2013-10-13
    所以情绪的周模式，日模式和小时模式是比较有分析价值的
    如果设定为分钟模式，则显得过于细碎，不能较好的体现出情绪变化的趋势
    所以在本函数中主要选用week, day, hour模式进行分析
    '''
    def take(vector):
        return vector[1]

    Matrix.sort(key=take)  # 按照时间对矩阵中的向量进行排序
    output_matrix = []  # 输出结果的字符串矩阵

    time_range = [Matrix[0][1], Matrix[-1][1]]
    emotion_list = ['anger', 'disgust', 'fear', 'joy', 'sadness']

    if mood in emotion_list:

        index = emotion_list.index(mood)

        mode_range_min = time_range[0]
        mode_range_max = time_range[1]

        if mode == 'week':
            emotion_sum = []
            for item in Matrix:
                emotion_sum.append(item[0][index])
            output = 'From 2013-10-11 to 2013-10-13, the ' + \
                emotion_list[index] + \
                ' proportion is %.6f' % (sum(emotion_sum)/len(emotion_sum))
            output_matrix.append(output)

        elif mode == 'day':

            for i in range(int(mode_range_min.tm_mday),
int(mode_range_max.tm_mday)+1):
                emotion_sum = []
                for item in Matrix:
                    if item[1].tm_mday == i:
                        emotion_sum.append(item[0][index])
                    else:
                        pass
                try:
                    output = "The 2013-10-%d\'s " % i + emotion_list[index] + '
proportion is %.6f' % (sum(
                        emotion_sum)/len(emotion_sum))
                    output_matrix.append(output)
                except:
                    break

        elif mode == 'hour':
            for i in range(int(mode_range_min.tm_mday),
int(mode_range_max.tm_mday)+1):
                for j in range(0, 24):
                    emotion_sum = []
                    for item in Matrix:
                        if item[1].tm_hour == j and item[1].tm_mday == i:
                            emotion_sum.append(item[0][index])
                        else:
                            pass
                    try:
                        output = "The 2013-10-%d %d:00--%d:00\'s " % (
                            i, j, j+1)+emotion_list[index]+' proportion is %.6f'
% (sum(emotion_sum)/len(emotion_sum))
                        output_matrix.append(output)
                    except:
                        break
        else:
            print("The time mode you choose is invalid")

    else:
        print("The emotion you choose is invalid.\n")

    return output_matrix


def emotion_distance(Matrix, mood, radius):
    '''
    分析情绪的空间分布

    实现一个函数可以通过参数来控制返回情绪的空间分布
    围绕某个中心点，随着半径增加，该情绪所占比例的变化(以经纬度的0.01为步长)
    而radius在输入时应该是0~0.09
    （在本方法中，中心点默认是城市的中心位置）
    '''

    position_x = []
    position_y = []

    output_matrix = []
    for i in Matrix:
        position_x.append(i[2][0])
        position_y.append(i[2][1])

    center_x = sum(position_x)/len(position_x)
    center_y = sum(position_y)/len(position_y)

    coordinates = np.array([center_x, center_y])

    for i in Matrix:
        vector = np.array(i[2])
        dis = np.linalg.norm(vector-coordinates)
        i.append(dis)

    def taken(item):
        return item[3]
    Matrix.sort(key=taken)

    emotion_list = ['anger', 'disgust', 'fear', 'joy', 'sadness']

    if mood in emotion_list:
        index = emotion_list.index(mood)
        emotion_dic = {}
        # for j in range(0.01, round(radius, 2), 0.01):
        for j in np.arange(0.01, round(radius+0.01, 2), 0.01):
            if j <= radius:
                j = round(j, 2)
                emotion_num = []
                for i in Matrix:
                    dis = i[3]
                    if j-0.01 <= dis:
                        if dis <= j:
                            emotion_num.append(i[0][index])
                        else:
                            break
                    else:
                        pass
            else:
                break

            try:
                emotion_dic[str(round(j-0.01, 2))+'~'+str(round(j, 2))
                            ] = sum(emotion_num)/len(emotion_num)
            except:
                print("Fail")

    else:
        print("The emotion you choose is invalid.\n")

    for i in emotion_dic.keys():
        output = "The proportion of %s in the %s area is %.6f." % (
            mood, i, emotion_dic[i])
        output_matrix.append(output)

    return output_matrix


if __name__ == '__main__':

    Matrix = []

    jieba_set_config()

    file = "C:/Users/DELL/Desktop/Code/BUAA_21/Week3/weibo.txt"

    clean_data = clear_data(file)

    func = analysis1(clean_data)
    Matrix = func(Matrix)

    output = emotion_trend(Matrix, 'sadness', 'day')

    output = emotion_trend(Matrix, 'joy', 'day')
    with open("C:/Users/DELL/Desktop/Code/BUAA_21/Week3/output.txt", 'a+') as f:

        output = emotion_trend(Matrix, 'sadness', 'day')

        print(*output, sep='\n', file=f)

        print('\n\n-----------------------------------------\n\n', file=f)

        output = emotion_trend(Matrix, 'joy', 'day')

        print(*output, sep='\n', file=f)

        print('\n\n-----------------------------------------\n\n', file=f)

        output = emotion_trend(Matrix, "joy", "hour")

        print(*output, sep='\n', file=f)

        print('\n\n-----------------------------------------\n\n', file=f)

        output = emotion_trend(Matrix, "joy", "week")

        print(*output, sep='\n', file=f)

        print('\n\n-----------------------------------------\n\n', file=f)

        # emotion_distance(Matrix, "joy", 0.05)

        output2 = emotion_distance(Matrix, "sadness", 0.07)

        print(*output2, sep='\n', file=f)

        print('\n\n-----------------------------------------\n\n', file=f)

        output2 = emotion_distance(Matrix, "joy", 0.05)

        print(*output2, sep='\n', file=f)

        print('\n\n-----------------------------------------\n\n', file=f)
```

实现过程

在本程序中，我们首先使用jieba_set_config函数进行jieba模块的参数设置，包括添加用户词典以及设置停用词。【停用词使用了第二次作业中提供的停用词，因为在两次文本处理的任务中，与正文内容无关的中文词差别不大，所以将其直接应用在第三次作业中是有一定依据的】

其次，通过clear_data函数，对微博内容进行清洗，包括清除url，同时使用jieba.analyse模块进行分词处理。与此同时，将微博中包含的时间数据转化为struct_time类型，将最后用中括号形式表示的坐标进行提取，与分词后的正文内容、时间一同包装成一个“向量”，将weibo.txt文本转化成一个量化的矩阵。

```
#部分向量实例
[['忘带', '泰康人寿', '大厦', '培训'], time.struct_time(tm_year=2013, tm_mon=10,
tm_mday=11, tm_hour=9, tm_min=3, tm_sec=31, tm_wday=4, tm_yday=284,
tm_isdst=-1), [39.9060994533, 116.36196252]]
[['下雨', '回来', '北京'], time.struct_time(tm_year=2013, tm_mon=10, tm_mday=12,
tm_hour=23, tm_min=57, tm_sec=55, tm_wday=5, tm_yday=285, tm_isdst=-1),
[40.01411, 116.516]]
[['4B', 'first', 'second', '2B', 'third', 'fourth', 'SB', '青年'],
time.struct_time(tm_year=2013, tm_mon=10, tm_mday=12, tm_hour=18, tm_min=17,
tm_sec=15, tm_wday=5, tm_yday=285, tm_isdst=-1), [39.926601, 116.26608]]
[['301', '工作餐', '免费', '深圳', '北京'], time.struct_time(tm_year=2013,
tm_mon=10, tm_mday=12, tm_hour=18, tm_min=18, tm_sec=9, tm_wday=5, tm_yday=285,
tm_isdst=-1), [39.9056556726, 116.27720642]]
[['五十块', '一百块', '好看', '努力', '喜欢', '人民币', '设计', '不是', '没有'],
time.struct_time(tm_year=2013, tm_mon=10, tm_mday=11, tm_hour=10, tm_min=47,
tm_sec=41, tm_wday=4, tm_yday=284, tm_isdst=-1), [40.03062, 116.3473]]
[['好喝', '好几块', '果汁', '大帝', '伯伯', '大方'], time.struct_time(tm_year=2013,
tm_mon=10, tm_mday=11, tm_hour=21, tm_min=30, tm_sec=46, tm_wday=4, tm_yday=284,
tm_isdst=-1), [39.892162, 116.27855]]
[['晚安', '重阳节', '相随', '快乐', '明天'], time.struct_time(tm_year=2013,
tm_mon=10, tm_mday=12, tm_hour=23, tm_min=41, tm_sec=53, tm_wday=5, tm_yday=285,
tm_isdst=-1), [40.000576, 116.48052]]
[['暗爽'], time.struct_time(tm_year=2013, tm_mon=10, tm_mday=12, tm_hour=13,
tm_min=32, tm_sec=10, tm_wday=5, tm_yday=285, tm_isdst=-1), [39.90958,
116.40419]]
[['狼崽子', 'EXO', '12', '哈皮', '蛋蛋', '很棒', '嘻嘻', '生日', '一起', '已经'],
time.struct_time(tm_year=2013, tm_mon=10, tm_mday=11, tm_hour=20, tm_min=50,
tm_sec=34, tm_wday=4, tm_yday=284, tm_isdst=-1), [39.976712, 116.35577]]
[['玩儿命', '特么', '手机卡', '倒腾', '干嘛', '半天', '知道'],
time.struct_time(tm_year=2013, tm_mon=10, tm_mday=11, tm_hour=21, tm_min=56,
tm_sec=3, tm_wday=4, tm_yday=284, tm_isdst=-1), [39.991879, 116.25044]]
[['忘却', '太快', '快乐', '脚步', '身边', '有时', '简单', '也许'],
time.struct_time(tm_year=2013, tm_mon=10, tm_mday=11, tm_hour=23, tm_min=57,
tm_sec=0, tm_wday=4, tm_yday=284,
tm_isdst=-1), [39.999714, 116.38548]]
```

此后，利用analysis1函数，对返回的量化形式的矩阵进行分析，包括根据分词结果生成一个1*5的情绪向量，根据时间早晚对矩阵的行进行排序。

```
#部分向量示例（已经去除情绪向量为零向量的情况）
[[0.0, 0.0, 0.0, 1.0, 0.0], time.struct_time(tm_year=2013, tm_mon=10,
tm_mday=11, tm_hour=8, tm_min=40, tm_sec=28, tm_wday=4, tm_yday=284,
tm_isdst=-1), [39.976222, 116.43601]]
[[0.0, 0.2, 0.0, 0.6, 0.2], time.struct_time(tm_year=2013, tm_mon=10,
tm_mday=11, tm_hour=23, tm_min=38, tm_sec=47, tm_wday=4, tm_yday=284,
tm_isdst=-1), [39.838283, 116.25152]]
[[0.0, 0.0, 0.0, 1.0, 0.0], time.struct_time(tm_year=2013, tm_mon=10,
tm_mday=11, tm_hour=21, tm_min=4, tm_sec=47, tm_wday=4, tm_yday=284,
tm_isdst=-1), [39.882248, 116.3758]]
[[0.6666666666666666, 0.0, 0.0, 0.3333333333333333, 0.0],
time.struct_time(tm_year=2013, tm_mon=10, tm_mday=11, tm_hour=15, tm_min=27,
tm_sec=45, tm_wday=4, tm_yday=284, tm_isdst=-1), [39.994, 116.3817]]
```

而emotion_trend函数则对整个时间范围内某一特定情绪随时间的变化趋势进行了分析。返回随时间增长某种情绪在整个时间段内所占的比例。【可以通过参数mood选择五种情绪中的一种，通过mode参数选择返回结果的时间模式（按周、天或者小时模式返回）】

emotion_distance函数则用来讨论情绪的空间分布。在选定所有坐标的中心作为中心点后，返回某种情绪围绕中心点的变化。【可以通过mode参数选择五种情绪中的一种，通过radius给定进行情绪比例变化分析的范围半径（0~0.09)】

最后使用print函数将结果输出到名为output.txt的文本文档中。

输出结果

```
Setting Jieba module config.

Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\DELL\AppData\Local\Temp\jieba.cache
Loading model cost 1.023 seconds.
Prefix dict has been built successfully.
```

```
The 2013-10-11's sadness proportion is 0.187423
The 2013-10-12's sadness proportion is 0.182883
The 2013-10-13's sadness proportion is 0.225806


-----------------------------------------


The 2013-10-11's joy proportion is 0.628609
The 2013-10-12's joy proportion is 0.615695
The 2013-10-13's joy proportion is 0.532258


-----------------------------------------


The 2013-10-11 0:00--1:00's joy proportion is 0.484694
The 2013-10-11 1:00--2:00's joy proportion is 0.539502
The 2013-10-11 2:00--3:00's joy proportion is 0.510823
The 2013-10-11 3:00--4:00's joy proportion is 0.650000
The 2013-10-11 4:00--5:00's joy proportion is 0.571429
The 2013-10-11 5:00--6:00's joy proportion is 0.500000
The 2013-10-11 6:00--7:00's joy proportion is 0.406250
The 2013-10-11 7:00--8:00's joy proportion is 0.464706
The 2013-10-11 8:00--9:00's joy proportion is 0.528571
The 2013-10-11 9:00--10:00's joy proportion is 0.614079
The 2013-10-11 10:00--11:00's joy proportion is 0.616410
The 2013-10-11 11:00--12:00's joy proportion is 0.665580
The 2013-10-11 12:00--13:00's joy proportion is 0.591210
The 2013-10-11 13:00--14:00's joy proportion is 0.640937
The 2013-10-11 14:00--15:00's joy proportion is 0.710349
The 2013-10-11 15:00--16:00's joy proportion is 0.637963
The 2013-10-11 16:00--17:00's joy proportion is 0.594615
The 2013-10-11 17:00--18:00's joy proportion is 0.538793
The 2013-10-11 18:00--19:00's joy proportion is 0.645825
The 2013-10-11 19:00--20:00's joy proportion is 0.618695
The 2013-10-11 20:00--21:00's joy proportion is 0.690924
The 2013-10-11 21:00--22:00's joy proportion is 0.672285
The 2013-10-11 22:00--23:00's joy proportion is 0.758439
The 2013-10-11 23:00--24:00's joy proportion is 0.676003
The 2013-10-12 0:00--1:00's joy proportion is 0.752297
The 2013-10-12 1:00--2:00's joy proportion is 0.541880
The 2013-10-12 2:00--3:00's joy proportion is 0.479167
The 2013-10-12 3:00--4:00's joy proportion is 0.350000
The 2013-10-12 4:00--5:00's joy proportion is 0.607143
The 2013-10-12 5:00--6:00's joy proportion is 0.593750
The 2013-10-12 6:00--7:00's joy proportion is 0.633333
The 2013-10-12 7:00--8:00's joy proportion is 0.605769
The 2013-10-12 8:00--9:00's joy proportion is 0.548942
The 2013-10-12 9:00--10:00's joy proportion is 0.685000
The 2013-10-12 10:00--11:00's joy proportion is 0.669872
The 2013-10-12 11:00--12:00's joy proportion is 0.641546
The 2013-10-12 12:00--13:00's joy proportion is 0.635029
The 2013-10-12 13:00--14:00's joy proportion is 0.650833
The 2013-10-12 14:00--15:00's joy proportion is 0.674157
The 2013-10-12 15:00--16:00's joy proportion is 0.548942
The 2013-10-12 16:00--17:00's joy proportion is 0.580108
The 2013-10-12 17:00--18:00's joy proportion is 0.659487
The 2013-10-12 18:00--19:00's joy proportion is 0.461228
The 2013-10-12 19:00--20:00's joy proportion is 0.582447
The 2013-10-12 20:00--21:00's joy proportion is 0.642972
The 2013-10-12 21:00--22:00's joy proportion is 0.589433
The 2013-10-12 22:00--23:00's joy proportion is 0.543882
The 2013-10-12 23:00--24:00's joy proportion is 0.638754
The 2013-10-13 0:00--1:00's joy proportion is 0.532258


-----------------------------------------


From 2013-10-11 to 2013-10-13, the joy proportion is 0.624603


-----------------------------------------


The proportion of sadness in the 0.0~0.01 area is 0.345238.
The proportion of sadness in the 0.01~0.02 area is 0.196920.
The proportion of sadness in the 0.02~0.03 area is 0.194291.
The proportion of sadness in the 0.03~0.04 area is 0.168693.
The proportion of sadness in the 0.04~0.05 area is 0.209967.
The proportion of sadness in the 0.05~0.06 area is 0.188151.
The proportion of sadness in the 0.06~0.07 area is 0.192459.


-----------------------------------------


The proportion of joy in the 0.0~0.01 area is 0.601190.
The proportion of joy in the 0.01~0.02 area is 0.659240.
The proportion of joy in the 0.02~0.03 area is 0.609027.
The proportion of joy in the 0.03~0.04 area is 0.599680.
The proportion of joy in the 0.04~0.05 area is 0.610446.


-----------------------------------------
```

字典方法进行情绪理解的优缺点

优点

以字典方法进行情绪理解，很大程度上依赖jieba库的分词结果。将一整句话切分为词组，降低了情绪分析的困难程度。将分词后的结果与情绪字典进行比较，结果也比较精确。

缺点

不能排除有些情况，网友可能“说反话”，在反问句式中用某些情绪词表达相反的态度（阴阳怪气）。比如：”你家哥哥真是太厉害了，上天入地无所不能，这么厉害怎么不去拯救世界呢？“此时就有可能出现错误的情绪理解的现象。

此外，情绪词典还存在着可以扩充的空间，比如在当今的网络用语中，会出现很多外文单词的中文表达（斯国一，一颗赛艇），但是并不能在jieba分词过程中被识别，就需要我们增加情绪词典中的词汇量，以提高情绪识别的精度。

总结

在进行数据情绪和处理中，用到了一部分正则表达式的知识，但是由于掌握的还不够充分，以至于处理的过程十分繁琐，走了很多不必要的弯路。但是因为个人对正则表达式的理解还不够深入，所以目前还没有参透改进提升的方法、

与此同时，在生成情绪向量时，连续几个if-elif语句并列，工作效率比较低；多次比较也造成了时间上的浪费。

程序中的循环语句太多，很多处理需要遍历进行，大大增加了时间成本。在本次作业中因数据量小所以体现的并不明显，但是一旦数据量增加到一定程度，就会出现程序运行缓慢的现象。

作业代码&输出文件

本次作业使用到的代码文件以及输出的文档均已上传至北航云盘以及个人Github代码库