20377199 赵芮箐 第3周作业

情绪理解是文本处理里最常见任务之一。现提供一个五类情绪字典（由情绪词组成，5个文件，人工标注），实现一个情绪分析工具，并利用该工具对10000条新浪微博进行测试和分析（一行一条微博）。微博数据见课程资料中提供的weibo.txt（200万条，包括地理位置，文本和发布时间），字典数据见公开数据中的emotion lexicon（https://doi.org/10.6084/m9.figshare.12163569.v2）。请按要求用函数进行功能封装，并在main中调用测试，鼓励尝试不同方式的可视化。

> 注意：如果规模太大处理不了，可以酌情处理一部分。

任务一：数据清洗

1. 实现一个函数，对微博数据进行清洗，去除噪声（如url等），过滤停用词。注意分词的时候应该将情绪词典加入jieba或pyltp的自定义词典，以提高这些情绪词的识别能力。

```
def cut_words(sen, stopwords_list):
    """
    分词函数：给出待分词语句，使用 jieba 分词，返回分词列表
    """
    wordlist = []
    for word in jieba.cut(sen, cut_all=False):
        if word not in stopwords_list:
            wordlist.append(word)
    return wordlist
```

```
def clean_text(filename):
    """
    清洗文件函数：清洗weibo评论数据，分词，将对应数据存成dataframe
    """
    emotion = ['anger.txt','disgust.txt','fear.txt','joy.txt','sadness.txt']
    path = ".\Anger makes fake news viral online-data&code\\data\\emotion_lexicon\\"
    for i in range(len(emotion)):                #将情绪词典加入jieba分词库
        jieba.load_userdict(os.path.normpath(path + emotion[i]))

    with open("D:\code\mp2022\week1\stopwords_list.txt", 'r', encoding='utf-8') as s:
        stopwords = s.read()
        stopwords_list = stopwords.split('\n')
```

```
    text_cut_list = []
    with open(filename,'r',encoding='utf-8') as f:
        print("----------正在清洗数据，请稍后----------")
        data = pd.read_csv(f, delimiter="\t")[1000000:]           # 读取文件存成df
        data.dropna()                                  # 补缺失值
        data.drop_duplicates(inplace=True)             # 去重复
        for i in range(len(data)):
            text = data.iloc[i,1]                      # 读取微博text
            #清洗微博
            text = re.sub(r"(回复)?(//)?\s*@\S*?\s*(:| |$)", " ", text)  # 去除正文中的@和回复/转发中的用户名
            #text = re.sub(r"\[\S+?]", "", text)         # 去除表情符号
            text = re.sub(r"#", "", text)              # 保留话题内容
            re.sub(r'http[:.]+\S+', '', text)          # 除去url
            text = re.sub(r"\s+", " ", text)           # 合并正文中过多的空格
            data.iloc[i,1] = text                      # 把清洗后的结果存进去
            text_cut = cut_words(text, stopwords_list)
            text_cut_list.append(text_cut)             # 将分词后的微博存成列表
        data["text_cut"] = text_cut_list               # 把分词后的微博存进df
    return data
```

部分结果展示：

```
----------正在清洗数据，请稍后----------
原微博为：今夜到明天白天有点想你，预计下午将持续想你，受延长低情绪影响，傍晚将转为大到暴想，心情由此将降低五度，预计此类天气将持续到见到你为止。@Sandy-英俊哥 米修 搜麻奇～～[亲亲][亲亲][亲亲] 幸好还有酸龟龟陪我[悲伤] 我在:http://t.cn/zRG9LQA
清晰后为：今夜到明天白天有点想你，预计下午将持续想你，受延长低情绪影响，傍晚将转为大到暴想，心情由此将降低五度，预计此类天气将持续到见到你为止。 米修 搜麻奇～～[亲亲][亲亲][亲亲] 幸好还有酸龟龟陪我[悲伤]
```

去除了@的用户对象、url等，把表情保留了，因为表情很多都含有情绪，也在情绪词典中

任务二：情绪分析

2. 实现两个函数，实现一条微博的情绪分析，返其情绪向量或情绪值。目前有两种方法，一是认为一条微博的情绪是混合的，即一共有n个情绪词，如果joy有n_1个，则joy的比例是n_1/n；二是认为一条微博的情绪是唯一的，即n个情绪词里，anger的情绪词最多，则该微博的情绪应该为angry。

> 注意，这里要求用闭包实现，尤其是要利用闭包实现一次加载情绪词典且局部变量持久化的特点。同时，也要注意考虑一些特别的情况，如无情绪词出现，不同情绪的情绪词出现数目一样等，并予以处理（如定义为无情绪，便于在后面的分析中去除）。

- 法一：认为一条微博的情绪是混合的，即一共有n个情绪词，如果joy有n_1个，则joy的比例是n_1/n

```
def emo_vector():
    print("----------正在分析情绪，请稍后----------")
    emodict = []
    path = ".\Anger makes fake news viral online-data&code\\data\\emotion_lexicon\\"
    filename = ['anger.txt','disgust.txt','fear.txt','joy.txt','sadness.txt']
    for i in range(len(filename)):
        file = open(os.path.normpath(path + filename[i]),'r',encoding='utf-8')
        emodict.append([line.strip() for line in file.readlines()])
        file.close()

    def count(text_cut):
        emo_count = [0, 0, 0, 0, 0]
        for word in text_cut:
            if word in emodict[0]:
                emo_count[0] += 1
            if word in emodict[1]:
                emo_count[1] += 1
            if word in emodict[2]:
                emo_count[2] += 1
            if word in emodict[3]:
                emo_count[3] += 1
            if word in emodict[4]:
                emo_count[4] += 1
        emo_sum = sum(emo_count)
        if emo_sum == 0:                                    # 无情绪时向量均为0
            emotion_vector = [0, 0, 0, 0, 0]
        else:
            emotion_vector = [i/sum(emo_count) for i in emo_count]
        return emotion_vector
    return count
```

部分结果示例：

```
----------正在分析情绪，请稍后----------
0                    也许生活就像一本掉了页的黄历，没有人会把它粘好并一页一页的回忆。
1    今夜到明天白天有点想你，预计下午将持续想你，受延长低情绪影响，傍晚将转为大到暴想，心情由此将...
2                          [抓狂][抓狂]像跟个定时炸弹聊天[衰][衰][衰]
3    【垂直电商低成本运营的三路九招】中国垂直电商逐步开始从规模导向转为利润导向。电商都在节衣缩食...
4                       我用智行火车票,成功秒杀10月08日北京西-广州东的硬座。
5    我们都有看不开的时候，总有冷落自己的举动，但是我我一定会提醒自己，如果还有明天，我们都有伤心...
6    翻了翻以前的照片，想了想以前的事儿。笑过哭过，吵过闹过，爱过恨过。其实，挺美好的，曾经，有……...
7    【易迅免费送iphone5c】免费iphone5c来啦，易迅无线惊喜无限，iphone5c免...
8                    [爱你]王诗龄和kimi。。。有没有口水流干了死翘的啊[可怜]我快了
9    不行了≪爸爸去哪儿≫这节目看不下去了。。。超想生孩子。我先歪沙发里假装睡着了听一耳朵吧[抓狂...
Name: text, dtype: object
0                                    [0, 0, 0, 0, 0]
1                            [0.0, 0.0, 0.0, 0.0, 1.0]
2                            [1.0, 0.0, 0.0, 0.0, 0.0]
3                            [0.0, 0.0, 0.0, 1.0, 0.0]
4                            [0.0, 0.0, 0.0, 1.0, 0.0]
5                            [0.0, 0.0, 0.0, 0.5, 0.5]
6                            [0.2, 0.2, 0.0, 0.6, 0.0]
7                            [0.0, 0.0, 0.0, 1.0, 0.0]
8    [0.0, 0.0, 0.0, 0.3333333333333333, 0.66666666...
9                            [0.4, 0.2, 0.0, 0.2, 0.2]
Name: text_emo, dtype: object
```

最后一句全文是：`不行了≪爸爸去哪儿≫这节目看不下去了。。。超想生孩子。我先歪沙发里假装睡着了听一耳朵吧[抓狂]好喜欢王诗龄和kimi`

打印了看了一下都是什么词汇影响的情绪：

```
sadness:不行
anger:看不下去
disgust:假装
anger:抓狂
joy:喜欢
```

震惊，居然是这些词汇！"不行"居然是sadness而不是anger，从这就感觉不是很好用了。

- 法二：认为一条微博的情绪是唯一的，即n个情绪词里，anger的情绪词最多，则该微博的情绪应该为angry

```python
def emo_kind():
    print("----------正在分析情绪，请稍后----------")
    emodict = []
    path = ".\Anger makes fake news viral online-data&code\\data\\emotion_lexicon\\"
    filename = ['anger.txt','disgust.txt','fear.txt','joy.txt','sadness.txt']
    for i in range(len(filename)):
        file = open(os.path.normpath(path + filename[i]),'r',encoding='utf-8')
        emodict.append([line.strip() for line in file.readlines()])
        file.close()

    def count(text_cut):
        emo_count = [0, 0, 0, 0, 0]
        for word in text_cut:
            if word in emodict[0]:
                emo_count[0] += 1
            if word in emodict[1]:
                emo_count[1] += 1
            if word in emodict[2]:
                emo_count[2] += 1
            if word in emodict[3]:
                emo_count[3] += 1
            if word in emodict[4]:
                emo_count[4] += 1
        emo_sum = sum(emo_count)
        emotion = ["anger", "disgust", "fear", "joy", "sadness"]
        if emo_sum == 0:                                    # 无情绪时返回种类none
            emotion_kind = "none"
        else:
            emotion_kind = emotion[emo_count.index(max(emo_count))]
        return emotion_kind
    return count
```

部分结果示例：

	text	text_emo
0	也许生活就像一本掉了页的黄历，没有人会把它粘好并一页一页的回忆。	none
1	今夜到明天白天有点想你，预计下午将持续想你，受延长低情绪影响，傍晚将转为大到暴想，心情由此将...	sadness
2	[抓狂][抓狂]像跟个定时炸弹聊天[衰][衰][衰]	anger
3	【垂直电商低成本运营的三路九招】中国垂直电商逐步开始从规模导向转为利润导向。电商都在节衣缩食...	joy
4	我用智行火车票,成功秒杀10月08日北京西-广州东的硬座。	joy
5	我们都有看不开的时候，总有冷落自己的举动，但是我我一定会提醒自己，如果还有明天，我们都有伤心...	joy
6	翻了翻以前的照片，想了想以前的事儿。笑过哭过，吵过闹过，爱过恨过。其实，挺美好的，曾经，有…...	joy
7	【易迅免费送iphone5c】免费iphone5c来啦，易迅无线惊喜无限，iphone5c免...	joy
8	[爱你]王诗龄和kimi。。。有没有口水流干了死翘的啊[可怜]我快了	sadness
9	不行了≪爸爸去哪儿≫这节目看不下去了。。。超想生孩子。我先歪沙发里假装睡着了听一耳朵吧[抓狂...	anger

情绪跟上文就是对应的！是情绪词最多对应的情绪！

5. （附加）讨论字典方法进行情绪理解的优缺点，有无可能进一步扩充字典来提高情绪识别的准确率？如何扩充，有无自动或半自动的扩充思路？

- 字典方法情绪理解的优缺点

  优点：方法比较简单易操作

  缺点：准确性很不高啊，纯评词语来判断情绪若遇到复杂的句式或者隐晦的情绪表达，很容易判断错误。

- 有无可能进一步扩充字典来提高情绪识别的准确率

  肯定会好很多，考虑更多可能性。但毕竟语言的文学性变化太复杂，只一味的扩充字典的话，提高的效果甚微，准确率应该还是很低

- 扩充的办法

会有半自动的方法，首先构建基础的情绪种子词，再用SO_PMI算法用语料库去训练它，即可帮我们扩充情绪字典，已经有很多人做这个项目了：https://blog.csdn.net/weixin_38008864/article/details/105172102

任务三：时空分析

3. 微博中包含时间，可以讨论不同时间情绪比例的变化趋势，实现一个函数，可以通过参数来控制并返回对应情绪的时间模式，如joy的小时模式，sadness的周模式等。

```
def emo_time(weibo,emotion,time):
    """
    情绪的时间分析函数：提供想要分析的情绪种类及时间模式,返回该情绪的变化趋势并绘制图形

    weibo:微博数据，储存为dataframe格式
    emotion:想分析的情绪
    time:想分析的时间模式

    """
    print("----------正在进行时间分析，请稍后----------")
    #先建好时间的字典
    hour = ['{:0>2d}'.format(i) for i in range(24)]
    hour_dict = {}
    hour_dict = hour_dict.fromkeys(hour,0)

    week = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']
    week_dict = {}
    week_dict = week_dict.fromkeys(week,0)

    month = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
    month_dict = {}
    month_dict = month_dict.fromkeys(month,0)
```

```
    time_list = np.array(weibo["weibo_created_at"]).tolist()    # 取得每条微博创建时间
    emotion_list = np.array(weibo["text_emo"]).tolist()         # 取得每条微博的情绪
    if time == 'hour':
        count = [1 for i in range(24)]
        for tm in time_list:
            t = tm.split(' ')
            t = t[-3].split(':')
            count[hour.index(t[0])] += 1
            if emotion_list[time_list.index(tm)] == emotion:
                hour_dict[t[0]] += 1
        hour_value = []
        for hr in hour:
            hour_value.append(hour_dict[hr]/count[hour.index(hr)])
        plt.plot(hour,hour_value,'o-',color='r',label='hour_{}'.format(emotion))
        plt.xlabel("hour")                              # 横坐标名字
        plt.ylabel("times")                             # 纵坐标名字
        plt.legend(loc = "best")                        # 图例
        for a,b in zip(hour,hour_value):
            plt.text(a,b+0.01,'%.2f' % (b),ha = 'center',va = 'bottom',fontsize=10)   # 标注点的数值
```

```
    elif time == 'week':
        count = [1 for i in range(7)]
        for tm in time_list:
            t = tm.split(' ')
            count[week.index(t[0])] += 1
            if emotion_list[time_list.index(tm)] == emotion:
                week_dict[t[0]] += 1
        week_value = []
        for wk in week:
            week_value.append(week_dict[wk]/count[week.index(wk)])
        plt.plot(week,week_value,'o-',color='b',label='week_{}'.format(emotion))
        plt.xlabel("week")                              # 横坐标名字
        plt.ylabel("times")                             # 纵坐标名字
        plt.legend(loc = "best")                        # 图例
        for a,b in zip(week,week_value):
            plt.text(a,b+0.01,'%.2f' % (b),ha = 'center',va = 'bottom',fontsize=10)   # 标注点的数值
```

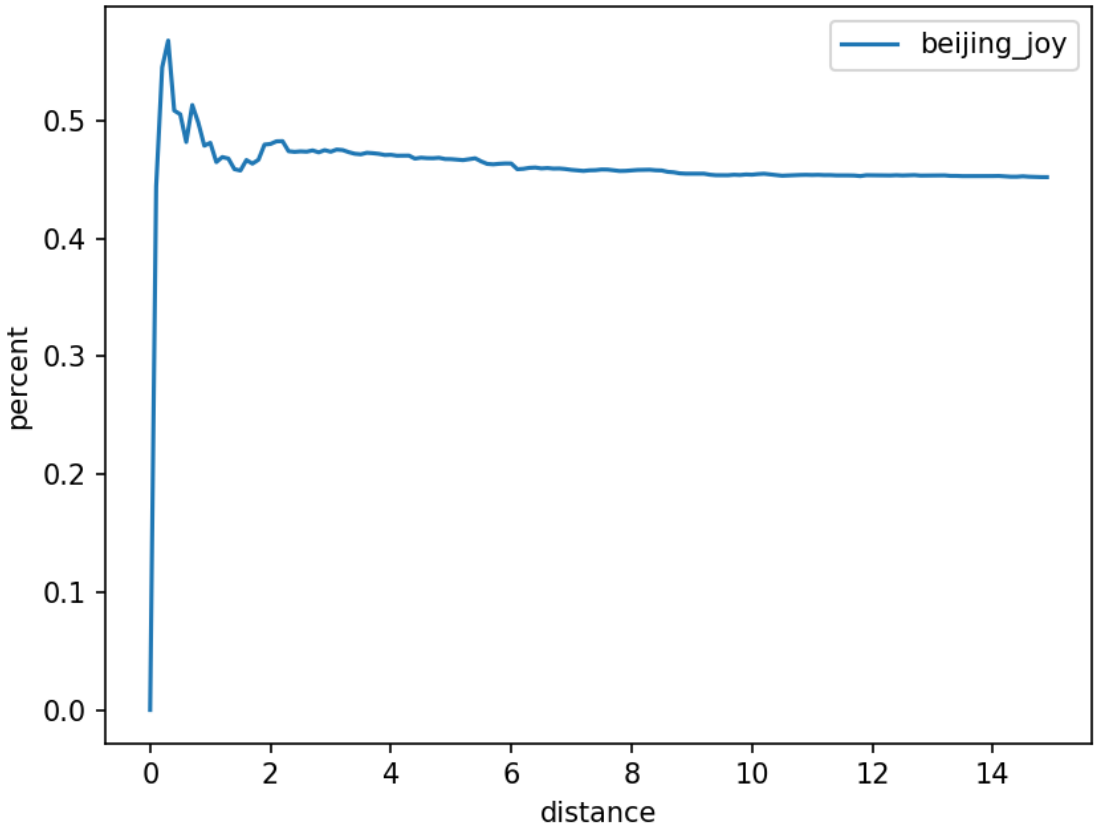
```
    elif time == 'month':
        count = [1 for i in range(12)]
        for tm in time_list:
            t = tm.split(' ')
            count[month.index(t[0])] += 1
            if emotion_list[time_list.index(tm)] == emotion:
                month_dict[t[1]] += 1
        month_value = []
        for mh in month:
            month_value.append(month_dict[mh]/count[month.index(mh)])
        plt.plot(month,month_value,'o-',color='y',label='month_{}'.format(emotion))
        plt.xlabel("month")                             # 横坐标名字
        plt.ylabel("times")                             # 纵坐标名字
        plt.legend(loc = "best")                        # 图例
        for a,b in zip(month,month_value):
            plt.text(a,b+0.01,'%.2f' % (b),ha = 'center',va = 'bottom',fontsize=10)   # 标注点的数值
    else:
        print('Error!')
    #plt.savefig('{}_{}.png'.format(time,emotion),dpi=800)
    plt.show()
```

4. 微博中包含空间，可以讨论情绪的空间分布，实现一个函数，可以通过参数来控制并返回对应情绪的空间分布，即围绕某个中心点，随着半径增加该情绪所占比例的变化，中心点可默认值可以是城市的中心位置。

```
def emo_area(weibo, emotion, city, max_dis):
    '''
    情绪空间分析函数：根据指定的情绪分析其空间的分布情况

    weibo:微博数据，储存为dataframe格式
    emotion：分析的情绪
    city：以该城市为中心
    max_dis：为计入的最远距离

    '''
    locations = str_to_float(np.array(weibo["location"]).tolist())    # 取得每条微博创建时间
    emotions = np.array(weibo["text_emo"]).tolist()             # 取得每条微博的情绪

    city_coord = {
    'beijing'  : [39.92, 116.46],
    'shanghai' : [31.22, 121.48],
    'guangzhou': [23.16, 113.23],
    'chengdu'  : [30.67, 104.06]}
    center = city_coord[city]                                   # 获取中心城市的坐标
```

```
    dist_list = []
    for i in range(len(emotions)):
        # 只统计中心周围经纬度小于1范围内的情绪
        if (abs(locations[i][0] - center[0]) < 1) and (abs(locations[i][1] - center[1]) < 1):
            dist = get_distance(center, locations[i])
            dist_list.append([dist,emotions[i]])
    dist_list = sorted(dist_list, key = (lambda x:x[0]))        # 按与中心的距离排序

    count = 1;cnt = 0
    percent = []
    x = list(np.arange(0, max_dis, 0.1))                        # 按差0.1公里分段统计
    for i in x:
        while dist_list[count][0] < i:
            if dist_list[count][1] == emotion:
                cnt += 1
            count += 1
        percent.append(cnt/count)
    plt.plot(x,percent,'o-',color='b',label='{}_{}'.format(city,emotion))
    plt.xlabel("distance")                          # 横坐标名字
    plt.ylabel("percent")                           # 纵坐标名字
    plt.legend(loc = "best")                        # 图例
    plt.show()
```

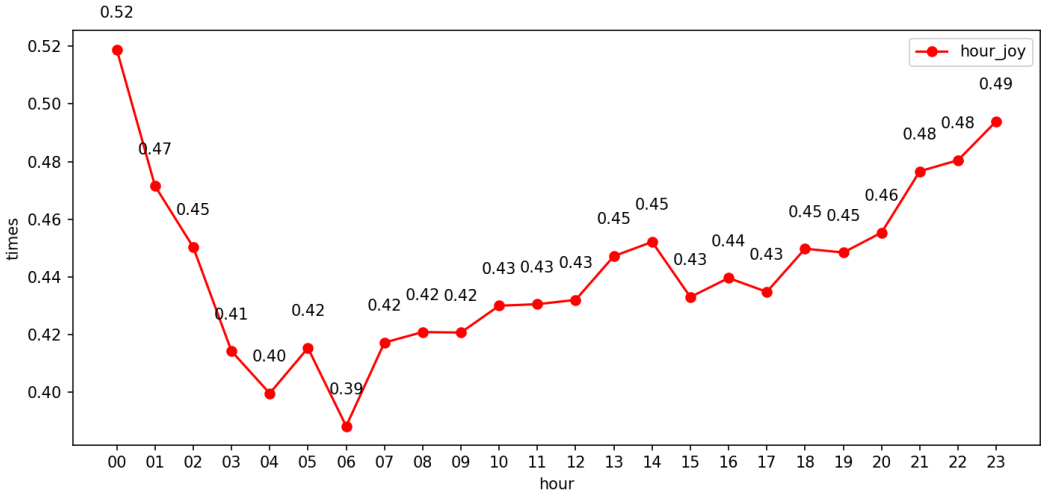
结果展示：



挺惊喜的发现当距离越远的时候，占比越趋于稳定，且趋于0.5（5个情绪中只有joy一个正面情绪，故占0.5）

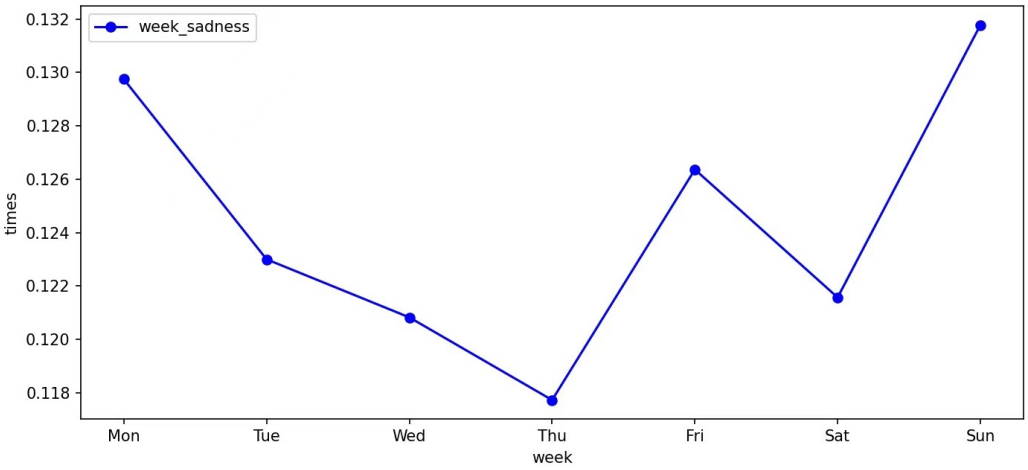
6. （附加）可否对情绪的时间和空间分布进行可视化？（如通过matplotlib绘制曲线，或者用pyecharts（注意版本的兼容性）在地图上标注不同的情绪）

- joy的小时分析



果然，现在网友都是到晚上上线娱乐，开心的情绪从5点（下班时间）开始晚上持续走高，零点达到峰值，然后下降。（睡觉的点+emo时刻到，不过冲浪的果然都是熬夜人）

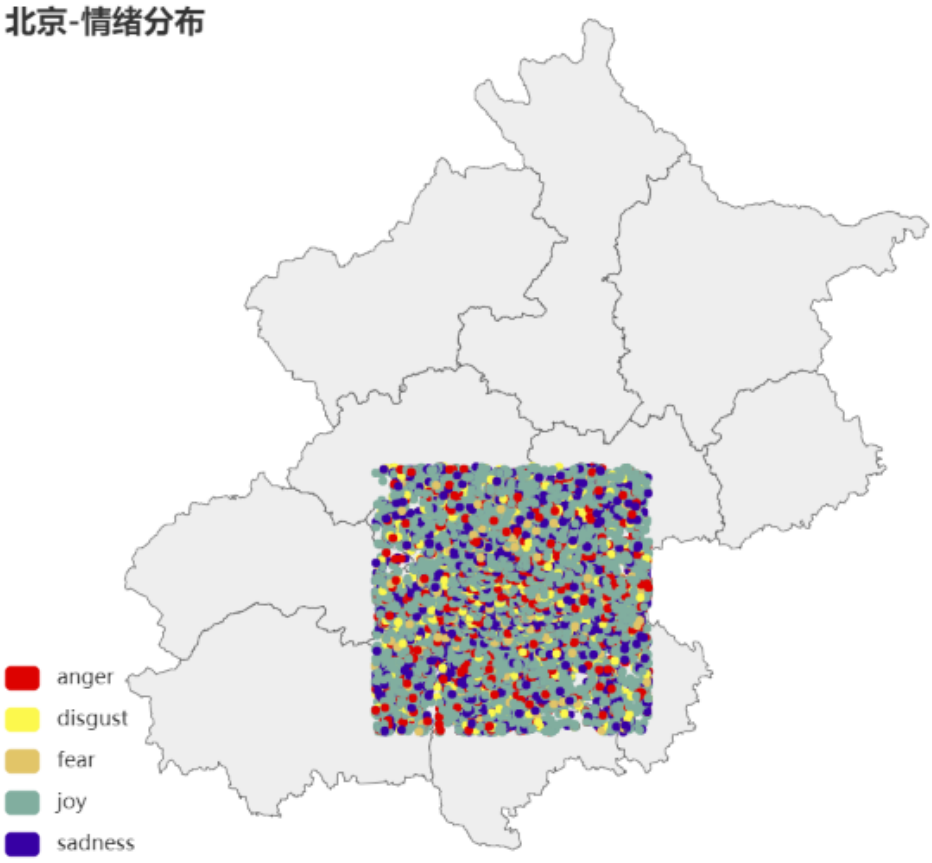
- sadness的周分析

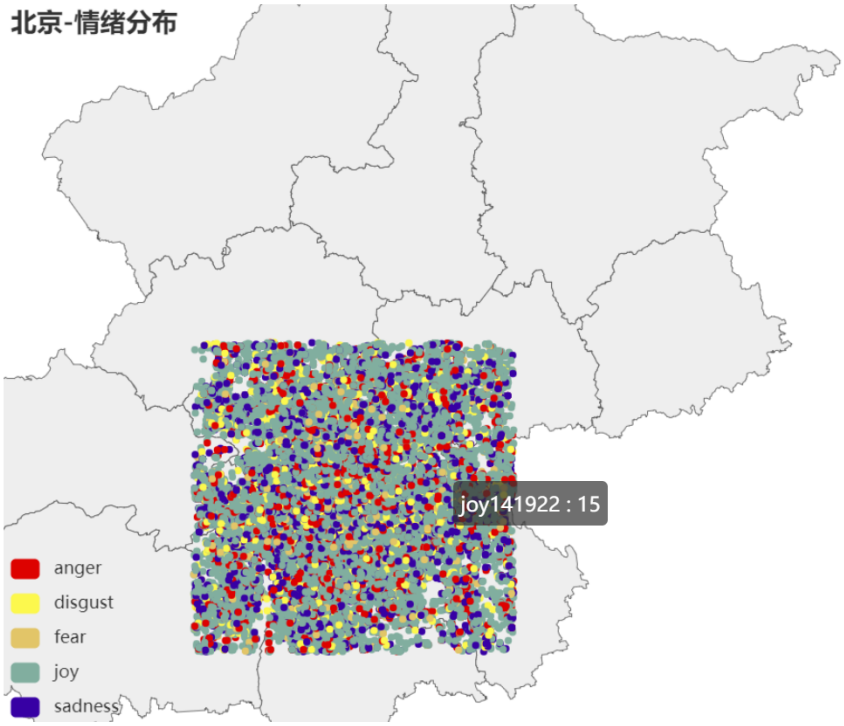


大家原来周四最不悲伤吗！！！因为疯狂星期四吗哈哈哈哈哈

周日和周一的悲伤情绪达到了极高点，果然是要上学/上班了，所以都要悲伤很多

- joy以北京为中心的空间分布





原来，数据就是取了这么一个方框！

7. （附加）思考情绪时空模式的管理意义，如营销等。

- 政府管理角度：
  对舆论控制管理很有帮助，能够快速判断和分析社会热点话题的爆发、传播的趋势，有助于分析其中可能存在的造谣带节奏等潜在风险隐患。
- 营销策略角度：
  对于精准投放广告等营销策略很有帮助，比如在joy时间峰值投放相关产品广告、开直播等，有助于销售。

代码：

https://github.com/rachhhhing/mp2022_python/blob/master/week3.py

Ref：

- 微博数据清洗：https://github.com/blmoistawinde/HarvestText
- 正则表达式：https://blog.csdn.net/jackandsnow/article/details/103885422
- Dataframe：https://www.runoob.com/pandas/pandas-dataframe.html
- matplotlib：https://blog.csdn.net/sinat_36219858/article/details/79800460
- pyecharts：https://blog.csdn.net/weixin_43746433/article/details/91346371