python week3

20377135 邹诺天来

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。

首先将情绪词典加入 jieba 自定义词典，并加载停用词文件：

```Python
#情绪词加入初始字典
def load():
    jieba.load_userdict(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\anger.txt")
    jieba.load_userdict(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\disgust.txt")
    jieba.load_userdict(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\fear.txt")
    jieba.load_userdict(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\joy.txt")
    jieba.load_userdict(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\sadness.txt")
```

```Python
# 创建停用词list
def stopwordslist(filepath):
    stopwords = [line.strip() for line in open(filepath, 'r',encoding='UTF-8').readlines()]
    return stopwords
```

定义多个函数，分别提取对应样本的地址、发布时月份、星期、时间（以小时为单位）以及内容，并储存在列表中。

位置信息通过提取每条样本字符串头部 '[]' 内内容获得：

```Python
```

```python
#提取位置信息
def findlocation(sentence):
    loc=[]
    for i in sentence:
        if i!=']':
            loc.append(i)
        if i==']':
            break
    loc.append(']')
    return ''.join(loc)
```

时间信息分为月份、星期、小时三种，其中月份与星期通过对关键字进行查找实现，小时通过查找月份与星期的位置进行实现：

Python

```python
#提取时间信息
def findtime(model,sentence):
    weeklist=['Mon','Tue','Wed','Thu','Fri','Sat','Sun']
    monthlist=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Se
p','Oct','Nov','Dec']
    if model=='week':
        for i in range(len(sentence)):
            if sentence[i:i+3] in weeklist:
                return sentence[i:i+3]
    if model=='month':
        for i in range(len(sentence)):
            if sentence[i:i+3] in monthlist:
                return sentence[i:i+3]
    if model=='time':
        for i in range(len(sentence)):
            #借助星期和月份共同判断，避免文本中出现星期或月份
            if (sentence[i:i+3] in weeklist) and (sentence[i+4:i+7
] in monthlist):
                return sentence[i+11:i+13]
```

文本内容信息通过提取删除位置后在星期前的所有字符实现：

Python

```python
#提取文本内容
def findcontent(sentence,location,week):
    sentence=sentence.replace(location,'')
    for i in range(len(sentence)):
        if sentence[i:i+3]==week:
            return sentence[:i]
```

通过正则函数清洗文本数据，删除 url：

```python
#清洗数据，删除url
def wash(sentence):
    text = re.sub("https*\S+", " ", sentence)
    return text
```

分词函数：

```python
# 对句子进行分词
def seg_sentence(sentence):
    sentence_seged = jieba.cut(sentence.strip())
    stopwords = stopwordslist('C:\\Users\\dexter\\Desktop\\stopwords_list.txt')  # 这里加载停用词的路径
    outstr = ''
    for word in sentence_seged:
        if word not in stopwords:
            if word != '\t':
                outstr += word
                outstr += " "
    return outstr
```

读取微博数据文件。由于文件过大，故仅对 1w 条微博进行情绪分析，为保证样本的随机性，从前 500w 条样本中借助 random 库进行随机抽取。

读取时使用 tqdm 添加进度条，查看实时进度。

```python
def add_item():
    inputs = open("C:\\Users\\dexter\\Desktop\\weibo.txt",encoding='UTF-8') #加载要处理的文件的路径
    load()
    location,week,month,time,content,emo=[],[],[],[],[],[]
    i=0
    num = random.sample(range(2000000), 10000)
    pbar = tqdm(total=10000)
    for item in inputs:
        if i in num:
            location.append(str(findlocation(item)))
            week.append(str(findtime('week',item)))
```

```
            month.append(str(findtime('month',item)))
            time.append(str(findtime('time', item)))
            con=findcontent(item,str(findlocation(item)),str(findt
ime('week',item)))
            con = wash(str(con))
            con=seg_sentence(con)
            content.append(con)
            emo.append(emotionlist(con)[1][0])
            pbar.update(1)
        i+=1
    pbar.close()
    return location,week,month,time,content,emo
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\dexter\AppData\Local\Temp\jieba.cache
Loading model cost 0.441 seconds.
Prefix dict has been built successfully.
  9%|▉         | 864/10000 [00:37<06:34, 23.15it/s]
```

实时进度条

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 n1 个，则 joy 的比例是 n1/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）。

通过闭包实现两种对情绪分析的方法，方法一返回文本中每个情绪所占比例，方法二则返回占比最大情绪及其占比（借助其在问题 1 'add_item' 中得到每条文本对应主要情绪）：

（当文本中没有任何词在情绪词典中，方法二返回 -1，在后续判断中通过该数值将无情绪文本进行剔除。）

Python

```
#两种方式计算情绪向量
def emotionlist(sentence):
    #加载情绪词典，方便后续判断情绪
```

```
    angry = [line.strip() for line in open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\anger.txt", 'r',encoding='UTF-8').readlines()]
    disgust = [line.strip() for line in open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\disgust.txt", 'r',encoding='UTF-8').readlines()]
    fear = [line.strip() for line in open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\fear.txt", 'r',encoding='UTF-8').readlines()]
    joy =  [line.strip() for line in open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\joy.txt", 'r',encoding='UTF-8').readlines()]
    sadness = [line.strip() for line in open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\sadness.txt", 'r',encoding='UTF-8').readlines()]

    def vector_1(sentence):
        nonlocal angry, disgust, fear, joy, sadness
        word = sentence.split()
        num = np.array([0, 0, 0, 0, 0])
        for item in word:
            if item in angry:
                num[0] += 1
                continue
            if item in disgust:
                num[1] += 1
                continue
            if item in fear:
                num[2] += 1
                continue
            if item in joy:
                num[3] += 1
                continue
            if item in sadness:
                num[4] += 1
                continue
        n = sum(num)
        if n==0:
            return 0
        return num / n,

    def vector_2(sentence):
        nonlocal angry, disgust, fear, joy, sadness
        word = sentence.split()
```

```
        num = [0, 0, 0, 0, 0]
        for item in word:
            if item in angry:
                num[0] += 1
                continue
            if item in disgust:
                num[1] += 1
                continue
            if item in fear:
                num[2] += 1
                continue
            if item in joy:
                num[3] += 1
                continue
            if item in sadness:
                num[4] += 1
                continue
        if max(num)==0:
            return [-1,0]
        return [np.argmax(num), max(num)]
    vector1=vector_1(sentence)
    vector2=vector_2(sentence)
    return [vector1,vector2]
```

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。

借助字典实现情绪的不同情况时间模式：

Python

```
#计算每个时间段不同情绪分布
def count_emotion(content,time,model):
    if model=='time':
        count = [[0 for i in range(5)] for i in range(24)]
        for i in range(len(content)):
            emo=emotionlist(content[i])
            #dic={0:'angry',1:'disgust',2:'fear',3:'joy',4:'sadnes
s'}
            if emo[1][0]==-1:
                continue
```

```
            count[int(time[i])][emo[1][0]]+=1
    if model=='week':
        count = [[0 for i in range(5)] for i in range(7)]
        for i in range(len(content)):
            emo=emotionlist(content[i])
            dic={'Mon':0,'Tue':1,'Wed':2,'Thu':3,'Fri':4,'Sat':5,
'Sun':6}
            if emo[1][0]==-1:
                continue
            count[dic[time[i]]][emo[1][0]]+=1
    if model=='month':
        count = [[0 for i in range(5)] for i in range(12)]
        for i in range(len(content)):
            emo=emotionlist(content[i])
            dic={'Jan':0,'Feb':1,'Mar':2,'Apr':3,'May':4,'Jun':5,
'Jul':6,'Aug':7,'Sep':8,'Oct':9,'Nov':10,'Dec':11}
            if emo[1][0]==-1:
                continue
            count[dic[time[i]]][emo[1][0]]+=1
    #print(count)
    return count
```

计算每个时间段内不同情绪的占比以及主要情绪：

Python

```
#计算每个时间主要情绪及其占比
def time_emo(count):
    dic = {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'joy', 4: 'sadn
ess'}
    cou=[[]for i in range(len(count))]
    for i in range(len(count)):
        index=np.argmax(count[i])
        if sum(count[i])==0:
            num=0
            cou[i]=['None',0]
        else:
            num=max(count[i])/sum(count[i])
            cou[i]=[dic[index],num]
    return cou
```

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。
首先将每个位置信息由字符串转化为列表形式：

```python
#将位置信息转化为列表形式
def loc_lis(location):
    lis_loc=[]
    for i in location:
        lis_loc.append(i[1:-1].split(', '))
    return lis_loc
```

计算在距离样本总中心给定范围内的各个情绪分布占比

```python
#计算随着与中心的距离变化的情绪分布
def loc_dis(loc,dis,emo):
    #计算中心
    x,y=0,0
    for i in loc:
        x+=float(i[0])
        y+=float(i[1])
    #print(x/len(loc),y/len(loc))
    def count_dis(dis):
        nonlocal x,y,loc,emo

        lis=np.array([0, 0, 0, 0, 0])
        for i in range(len(loc)):
            if math.sqrt((float(loc[i][0])-x/len(loc))**2+(float(l
oc[i][1])-y/len(loc))**2) < dis and emo[i]!=-1:
                lis[emo[i]]+=1
        if sum(lis[i] for i in range(5))==0:
            return lis
        return lis/sum(lis[i] for i in range(5))
    return count_dis(dis)
```

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？

优点：通过识别字典中特定词语对情绪的判断能达到较为准确的程度，且能得到各种情绪极为精确的比例；

缺点：各种情绪在分布较为均匀的情况下对文本的主要情绪判断会产生困难，同

时，近年来微博越来越多的人说话开始阴阳怪气，一些本用于表达开心等的情绪词如今再出现可能表达的情绪与其本意大相径庭，导致判断的情绪方向完全错误。（比如说我自己。。。）

进一步扩充字典在一定程度上仍能对提高情绪识别的准确率起到一定作用，为实现自动的扩充，可以将文本判断时不属于情绪词的关键词加入在该文本中出现过的情绪的新建列表中，通过统计新列表中不同关键词出现的词频，以某个特定数值作为分界线，将词频大于该数值的关键词加入对应情绪的词典，以此实现自动扩充。

首先，得到句子的情绪构成，将不属于任何情绪的词语分别添加进在本句中出现过的情绪的列表：

Python

```python
#添加字典关键词
def add_word(sentence):
    #调用emotionlist,基于emotionlist结果再次遍历文本
    lis = emotionlist(sentence)[0]
    lis1, lis2, lis3, lis4, lis5 = [],[],[],[],[]
    # 加载情绪词典，方便后续判断情绪
    angry = [line.strip() for line in open(
        r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\anger.txt", 'r',
        encoding='UTF-8').readlines()]
    disgust = [line.strip() for line in open(
        r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\disgust.txt", 'r',
        encoding='UTF-8').readlines()]
    fear = [line.strip() for line in
            open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\fear.txt",
                 'r', encoding='UTF-8').readlines()]
    joy = [line.strip() for line in
           open(r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\joy.txt",
                'r', encoding='UTF-8').readlines()]
    sadness = [line.strip() for line in open(
        r"C:\Users\dexter\Desktop\Anger makes fake news viral online-data&code\data\emotion_lexicon\sadness.txt", 'r',
        encoding='UTF-8').readlines()]

    word = sentence.split()
    for item in word:
        if (item not in angry) and (item not in disgust) and (item not in fear) and (item not in joy)\
```

```
                and (item not in sadness):
            if lis[0]!=0:
                lis1.append(item)
                continue
            if lis[1]!=0:
                lis2.append(item)
                continue
            if lis[2]!=0:
                lis3.append(item)
                continue
            if lis[3]!=0:
                lis4.append(item)
                continue
            if lis[4]!=0:
                lis5.append(item)
    #lis1-5分别对应5种情绪
    return lis1, lis2, lis3, lis4, lis5
```

随后，遍历所有样本，通过字典对与情绪相关的每个无关词进行计数，对出现频次进行判断，定义如果一个词语与某种情绪在不同样本中同时出现超过二十次，判定其与该情绪有一定关系，加入列表，在稍后将列表写入情绪词典：

Python

```python
def add_dic(content):
    dic1,dic2,dic3,dic4,dic5={},{},{},{},{}
    for sentence in content:
        lis1, lis2, lis3, lis4, lis5 = add_word(sentence)
        def add(lis,dic):
            for item in lis:
                if item not in dic:
                    dic[item]=1
                else:
                    dic[item]+=1
            return dic
        dic1 = add(lis1,dic1)
        dic2 = add(lis2,dic2)
        dic3 = add(lis3,dic3)
        dic4 = add(lis4,dic4)
        dic5 = add(lis5,dic5)
    def travel(dic):
        lis=[]
        for i,j in dic.items():
            if j>20:
```

```
                lis.append(i)
        return lis
    lis1 = travel(dic1)
    lis2 = travel(dic2)
    lis3 = travel(dic3)
    lis4 = travel(dic4)
    lis5 = travel(dic5)
    return lis1, lis2, lis3, lis4, lis5
```

最后，遍历每个列表，将其中词语写入对应情绪词典：

```python
def add_file(lis,filename):
    file = open(filename, 'a',encoding='UTF-8')
    for item in lis:
        file.write(item + '\n')
    file.close()
```

该方法的不足之处在于一个句子可能存在多种情绪，将导致最后结果中一个词分别加入不同词典，且该词可能并不具有太多感情色彩，仅仅因为其出现频次高而被挑选出，如，在一次运行中，需要加入词典的结果如下图：

```
['孩子', '生活', '真的', '想', '吃', '做', '手机', '怒', '事', '发现', '北京', '走', '朋友', '真']
['真的', '做', '想', '中', '感觉', '生活', '北京', '工作', '挖', '朋友', '时间', '真', '找', '吃', '老人']
['中', '想', '北京', '做']
['孩子', '生活', '听说', '男人', '中', '一点', '真', '妈妈', '不用', '身边', '朋友', '买', '看着', '点', '终于',
['孩子', '生活', '中', '一点', '真', '真的', '明天', '事情', '睡不着', '身边', '朋友', '听', '找', '地方', '发现'
```

从上至下依次代表加入 'angry', 'disgust', 'fear', 'joy', 'sadness' 词典的词语

由上图来看，很明显在不同语境下“生活”、“孩子”等可能具有不同情绪，但同时“怒”、“睡不着”等也确实有一定指向性。

考虑将函数进行改进，在筛选时仅将无关词添加进该句主情绪所属列表中，考虑到因此频次将减少，将边界值定为 10。修改部分代码如下：

```python
#添加字典关键词
def add_word(sentence):
    #调用emotionlist,基于emotionlist结果再次遍历文本
    num = emotionlist(sentence)[1][0]
    lis1, lis2, lis3, lis4, lis5 = [],[],[],[],[]
    # 加载情绪词典，方便后续判断情绪
    angry = [line.strip() for line in open(
        r"C:\Users\dexter\Desktop\Anger makes fake news viral onli
```

```
ne-data&code\data\emotion_lexicon\anger.txt", 'r',
        encoding='UTF-8').readlines()]
    disgust = [line.strip() for line in open(
        r"C:\Users\dexter\Desktop\Anger makes fake news viral onli
ne-data&code\data\emotion_lexicon\disgust.txt", 'r',
        encoding='UTF-8').readlines()]
    fear = [line.strip() for line in
            open(r"C:\Users\dexter\Desktop\Anger makes fake news v
iral online-data&code\data\emotion_lexicon\fear.txt",
                 'r', encoding='UTF-8').readlines()]
    joy = [line.strip() for line in
           open(r"C:\Users\dexter\Desktop\Anger makes fake news vi
ral online-data&code\data\emotion_lexicon\joy.txt",
                'r', encoding='UTF-8').readlines()]
    sadness = [line.strip() for line in open(
        r"C:\Users\dexter\Desktop\Anger makes fake news viral onli
ne-data&code\data\emotion_lexicon\sadness.txt", 'r',
        encoding='UTF-8').readlines()]

    word = sentence.split()
    for item in word:
        if (item not in angry) and (item not in disgust) and (item
not in fear) and (item not in joy)\
                and (item not in sadness):
            if num==0:
                lis1.append(item)
                continue
            if num==1:
                lis2.append(item)
                continue
            if num==2:
                lis3.append(item)
                continue
            if num==3:
                lis4.append(item)
                continue
            if num==4:
                lis5.append(item)
    return lis1, lis2, lis3, lis4, lis5
```

改进后运行结果如下：

```
['里', '时间', '真', '朋友', '怒', '手机', '想', '不想', '北京', '真的', '走', '吃', '明天', '做', '车', '完']
['挖', '感觉', '事', '想', '找', '走', '中', '真的', '吃', '生活', '点', '做', '里', '北京', '真', '字']
```

从上至下仍依次代表加入 'angry', 'disgust', 'fear', 'joy', 'sadness' 词典的词语

可以看到在五个列表中同时出现的词语大大减少，虽然仍不是完全合理，但能让人感觉到具有一定情绪的词语有所提升。

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）

首先通过 matplotlib 实现对不同时间、距离样本中心不同距离的情绪分布的可视化：

Python

```python
#绘制随时间变化情绪分布折线图
def plot_time(count_emo,model,i):
    if model=='week':
        time = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']

    if model == 'time':
        time = [i for i in range(24)]

    if model == 'month':
        time = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','S
ep','Oct','Nov','Dec']
    angry = [i[0] for i in count_emo]
    disgust = [i[1] for i in count_emo]
    fear = [i[2] for i in count_emo]
    joy = [i[3] for i in count_emo]
    sadness = [i[4] for i in count_emo]
    plt.subplot(2,3,i)

    plt.xlabel(model)  # x轴名称 只能是英文
    plt.ylabel("frequency")  # y轴名称 只能是英文
    plt.title("emotion in different "+ model , fontdict={"fontsiz
e": 16})  # 添加标题，调整字符大小

    plt.plot(time, angry, 'r', label="angry")
    plt.plot(time, disgust, 'm', label="disgust")
    plt.plot(time, fear, 'k', label="fear")
    plt.plot(time, joy, 'g', label="joy")
    plt.plot(time, sadness, 'b', label="sadness")
```

```
    plt.legend()
```

Python

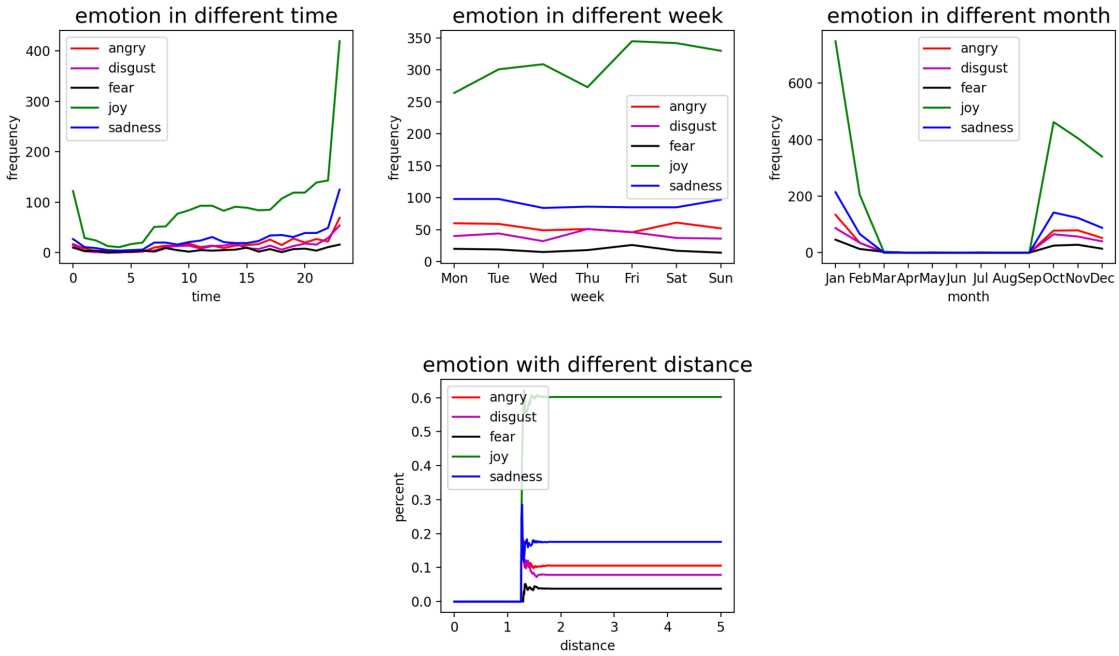
```python
#绘制随与中心的距离变化情绪占比的变化曲线图
def plot_dis(loc,emo,n):
    x = np.linspace(0, 5, 500)
    angry,disgust,fear,joy,sadness=[],[],[],[],[]
    for i in x:
        y = loc_dis(loc,i,emo)
        angry.append(y[0])
        disgust.append((y[1]))
        fear.append(y[2])
        joy.append(y[3])
        sadness.append(y[4])

    plt.subplot(2,3,n)
    plt.xlabel("distance")  # x轴名称 只能是英文
    plt.ylabel("percent")  # y轴名称 只能是英文
    plt.title("emotion with different distance", fontdict={"fontsi
ze": 16})  # 添加标题，调整字符大小
    plt.plot(x, angry,'r',label="angry")
    plt.plot(x, disgust,'m',label="disgust")
    plt.plot(x, fear,'k',label="fear")
    plt.plot(x,joy,'g',label="joy")
    plt.plot(x, sadness,'b',label="sadness")

    plt.legend()
```

本题中两个函数均没有添加 plt.show()，是由于想让四幅图最后同时显示，故将 plt.show()添加在主函数中该两函数执行之后，见下文

函数执行后所得可视化结果如下：



第一行从左至右依次为不同情绪按照 24h、1 周、12 个月为周期对情绪出现频次进行统计的可视化结果；第二行为以样本经纬度均值为中心统计随距离增长不同情绪所占百分比的可视化结果。

可以看出，在所抽样本中，'joy' 情绪出现的次数一骑绝尘，同时，22：00–0：00 时间段是一天中发博数量最多的时间段，结合现实，这段时间微博网友大部分已经下班或者放学，是一天中玩手机的高峰时间，说明了分析结果的合理性；样本中主要为 10 月至次年 2 月的数据，总的来说，情绪占比相差不大，说明情绪受月份影响不大，而 1 月 'joy' 数量上升可以理解为新一年的期待与祝福；从距离来看，distance=1 内几乎没有样本存在，从在 1–2 中间发生飙升，说明在此处开始出现样本，其后大致占比保持稳定，说明情绪受地区影响不大（至少在同城范围内）。

随后使用 pyecharts 在地图上进行情绪标注：

Python

```python
#返回每句话分析所得情绪
def count_all(content):
    dic = {-1:'None',0: 'angry', 1: 'disgust', 2: 'fear', 3: 'joy', 4
: 'sadness'}
    emo=[]
    for i in content:
        emotion = emotionlist(i)
        index = emotion[1][0]
        emo.append(dic[index])
    return emo
```

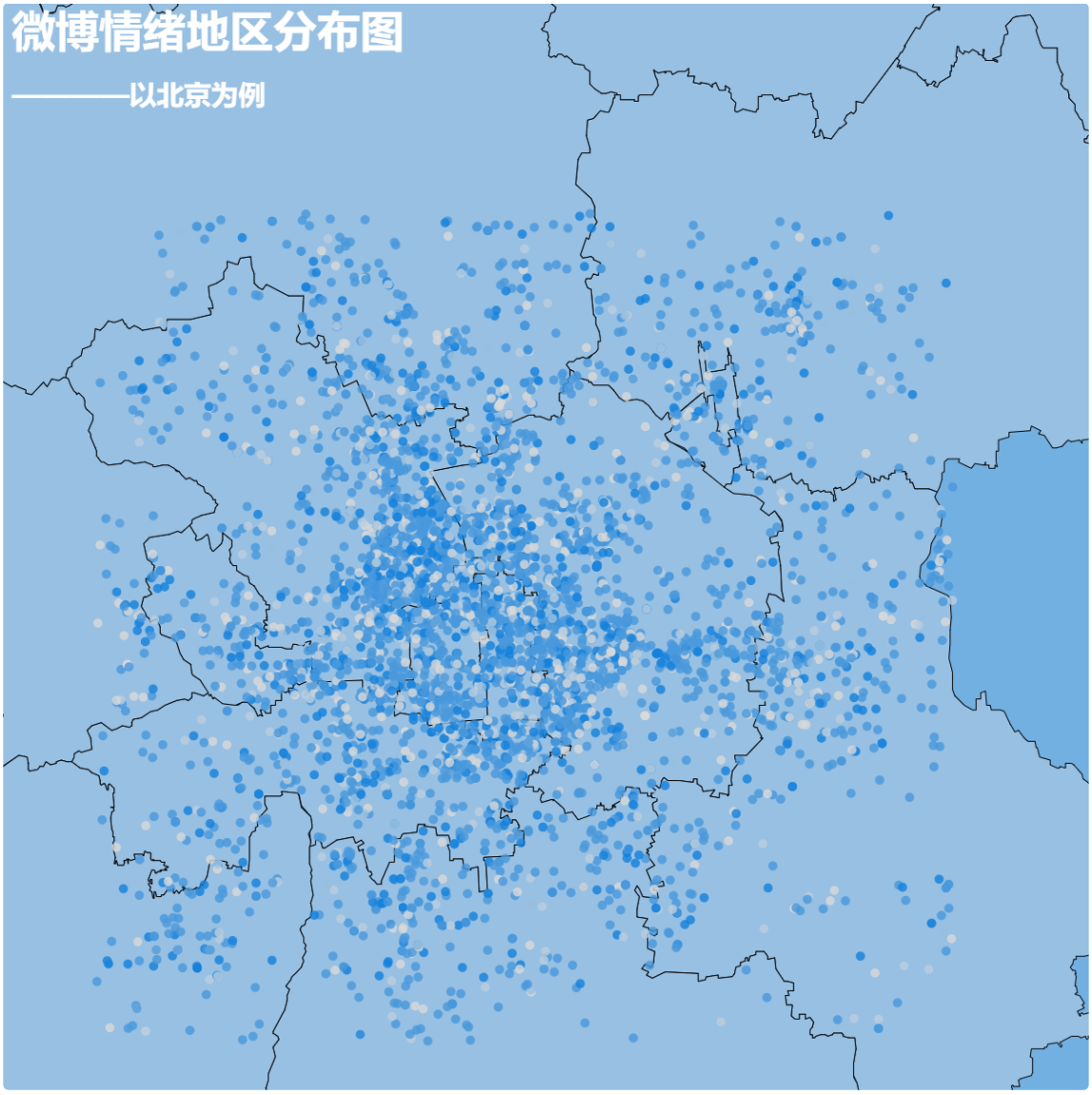
Python

```python
def test_geo(emo,loc):
    g = (
        Geo(
        init_opts=opts.InitOpts(width="600px", height="600px", page_title="map", bg_color='#73B0E2')
        # 颜色是str的16进制或英文都可以
    ).add_schema(
        maptype="北京",
        itemstyle_opts=opts.ItemStyleOpts(
            color="#97C0E3"  # 背景颜色
            , border_color="black")
        # 地图类型
         # 边界线颜色
    )
    )
    #lis1,lis2,lis3,lis4,lis5=[],[],[],[],[]
    dic = {"angry":'#DADADA',"disgust":'#BBCEDF','fear':'#9ABFDE','joy':'#4999DC','sadness':'#1381DC'}
    for i in range(len(emo)):
        if emo[i]=='None':
            continue
        g.add_coordinate(emo[i], loc[i][1], loc[i][0])
        g.add(
            series_name=i  # 系列名
            , data_pair=[(emo[i],i)]  # 系列里需要的点用列表框住多个元组达到批量输入的效果[(坐标点1，坐标点1的值),(坐标点2，坐标点2的值),(坐标点3，坐标点3的值)]
            , symbol_size=5  # 系列内显示点的大小
            , color=dic[emo[i]]
            , is_selected=True
        )

    g.set_series_opts(
        label_opts=opts.LabelOpts(
            is_show=False
        ))
    g.set_global_opts(
        title_opts=opts.TitleOpts(
            title='微博情绪地区分布图',  # 主标题内容
            subtitle='————以北京为例',  # 副标题内容
            item_gap=15,  # 主副标题的间距
            title_textstyle_opts=opts.TextStyleOpts(
```

```
                color="white",  # 主标题颜色
                font_weight="bolder",  # 主标题加粗
                font_size=24  # 主标题字体大小
            ),
            subtitle_textstyle_opts=opts.TextStyleOpts(
                color='white',  # 副标题颜色
                font_weight="bolder",  # 副标题加粗
                font_size=15))  # 副标题副标题字体大小
        , legend_opts=opts.LegendOpts(pos_right="10px", inactive_colo
r="white",
                                      textstyle_opts=opts.TextStyleOp
ts(color="orange"))
    )
    result = g.render()  # 会在你py文件同目录下生成一个html文件，也可在括号
内输入保存路径，用浏览器打开html文件可以查看
    webbrowser.open(result)
```

可视化结果如下：



图为以北京市地图为背景对微博样本的空间分布进行可视化的结果，将不同情绪按经纬度在地图上进行标记，不同情绪对应不同颜色标记点

7.（附加）思考情绪时空模式的管理意义，如营销等。

基于情绪词典方法判断情绪虽有一定准确率，但仍有很大的改进空间，后续可以结合机器学习相关算法对样本进行训练，得到更为优越的语义分析模型。但尽管参考性有限，该方法对现实应用仍有一定作用。

进行情绪时空模式管理，可以根据大数据对地区人群大致情绪状况进行分析，再根据分析结果进行推送，达到使用户产生产生共鸣的效果，按分析结果投送广告，也能提高营销效果；

同时，也可以使用户加强自我的情绪的认知。

最后附上 main()函数:

Python

```python
def main():
    location,week,month,time,content,emo = add_item()
    count_emo_t = count_emotion(content,time,'time')
    count_emo_w = count_emotion(content, week, 'week')
    count_emo_m = count_emotion(content, month, 'month')

    result_t = time_emo(count_emo_t)
    result_w = time_emo(count_emo_w)
    result_m = time_emo(count_emo_m)

    lis1, lis2, lis3, lis4, lis5 = add_dic(content)
    add_file(lis1,
             r"C:\Users\dexter\Desktop\Anger makes fake news viral on
line-data&code\data\emotion_lexicon\anger.txt")
    add_file(lis1,
             r"C:\Users\dexter\Desktop\Anger makes fake news viral on
line-data&code\data\emotion_lexicon\disgust.txt")
    add_file(lis1,
             r"C:\Users\dexter\Desktop\Anger makes fake news viral on
line-data&code\data\emotion_lexicon\fear.txt")
    add_file(lis1, r"C:\Users\dexter\Desktop\Anger makes fake news vi
ral online-data&code\data\emotion_lexicon\joy.txt")
    add_file(lis1,
             r"C:\Users\dexter\Desktop\Anger makes fake news viral on
line-data&code\data\emotion_lexicon\sadness.txt")

    plt.subplots_adjust(left=None, bottom=None, right=None, top=None,
wspace=0.3, hspace=0.5)
    plot_time(count_emo_t,'time',1)
    plot_time(count_emo_w, 'week',2)
    plot_time(count_emo_m, 'month',3)
    loc=loc_lis(location)
    plot_dis(loc,emo,5)
    plt.show()

    emo_ = count_all(content)
    test_geo(emo_,loc)
```

Python

```python
if __name__=='__main__':
```