

微博大 V 用户画像与热点话题分析

-----17271229 杨尽能

该文档包含设计思路、过程和分析结果，完整代码见ipynb文档。

一、查看数据

1、导入需要的包

```
import csv
import pandas as pd
import chardet
import jieba
import wordcloud
from PIL import Image
from jieba import analyse
import numpy as np
from pyecharts.charts import Pie
import pyecharts.options as opts
from pyecharts.charts import Map
import matplotlib.pyplot as plt
%matplotlib inline
#from pyecharts.globals import CurrentConfig, NotebookType
# CurrentConfig.NOTEBOOK_TYPE = NotebookType.JUPYTER_LAB
#CurrentConfig.ONLINE_HOST = "pyecharts-assets/assets"
```

2、用pandas导入数据并查看

```
user_df=pd.read_csv('data/data31204/userdata.csv')
user_df
```

	uid	province	gender	verified
0	uCXZAHQXC	11	m	False
1	uUPCCYCXC	11	f	False
2	u351ODTXW	11	f	False
3	uG1K5KFX5	11	m	False
4	u0VNQ2GX5	11	f	False
5	uMM9X7YQ	11	m	False
14388378	uVLEQA35TMI	11	m	False
14388379	uEMGO5KU2	13	m	False
14388380	uGL4WJVVRW	11	m	False
14388381	uVT4ROKNO	11	f	False
14388382	uSIIYS3GF	13	f	False
14388383	uMMMYRHGF	11	f	False
14388384	uUBIYAFZU	44	m	False

14388385 rows × 4 columns

这里直接读取week1.csv的话会有编码问题，所以要间接用open函数里的errors参数来处理。

```
data_path='data/data31204/week1.csv'
df = pd.DataFrame()
encode = get_encode(data_path) # get_encode函数在上文
f = open(data_path, encoding=encode,errors='ignore')
data = pd.read_csv(f,dtype=str)
df = df.append(data)
df
```

	mid	retweeted_status_mid	uid	retweeted_uid	source	image
0	mCCIUNCqwe	mU5j0dlAkQ	uK3RXUJ0V	NaN	新浪微博	0
1	mRsOcOLTlc	mJGNX5nAmo	uK3RXUJ0V	NaN	新浪微博	0
2	mH44qG6iUm	mH44qL9LIF	uK3RXUJ0V	NaN	新浪微博	0
3	mZmwFtOdVX	mcyE5GR7Gj	uK3RXUJ0V	NaN	新浪微博	0
4	mQkLJSI8bf	muy8VxftBB	uK3RXUJ0V	NaN	新浪微博	0
5	mnzrsoGWNN	mNfGcUeZbK	uK3RXUJ0V	NaN	新浪微博	0
6	m2rVkbmLsg	m7nJh3W6z	uK3RXUJ0V	NaN	新浪微博	0
7	mNfG6Xsbx5	mex2cwWppM	uK3RXUJ0V	NaN	新浪微博	0

	text	geo	created_at	deleted_last_seen	permission_denied
		NaN	2012-01-03 02:02:27	NaN	NaN
		NaN	2012-01-03 01:17:39	NaN	NaN
		NaN	2012-01-03 01:15:36	NaN	NaN
		NaN	2012-01-03 01:12:55	NaN	NaN
		NaN	2012-01-03 01:10:42	NaN	NaN
		NaN	2012-01-03 01:09:54	NaN	NaN
		NaN	2012-01-03 01:08:45	NaN	NaN
		NaN	2012-01-03 00:54:07	NaN	NaN

用info函数来查看两个数据表的信息，info函数返回有哪些列、有多少非缺失值、每列的类型。

df:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4790108 entries, 0 to 4790107
Data columns (total 11 columns):
mid                                object
retweeted_status_mid              object
uid                                object
retweeted_uid                      object
source                            object
image                             object
text                              object
geo                                object
created_at                        object
deleted_last_seen                 object
permission_denied                 object
dtypes: object(11)
memory usage: 402.0+ MB
```

user_df:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14388385 entries, 0 to 14388384
Data columns (total 4 columns):
uid          object
province     int64
gender       object
verified     bool
dtypes: bool(1), int64(1), object(2)
memory usage: 343.0+ MB
```

查看有多少是缺失值:

df:

```
df.isnull().sum()
```

```
mid          0
retweeted_status_mid  1619152
uid          0
retweeted_uid  4608000
source       52
image        0
text         70
geo          4768168
created_at   0
deleted_last_seen  4770468
permission_denied  4790099
dtype: int64
```

可见retweeted_uid、geo、deleted_last_seen、permission_denied基本上全是Nan值，所以不需要考虑他们。

user_df:

```
user_df.isnull().sum()
```

```
uid          0
province     0
gender       0
verified     0
dtype: int64
```

可见user_df里没有空值。

二、清洗数据

user_df的数据不需要清洗，来看df。

随便从text中抽一条看看：

```
'太萌了><//@uKPK1KLQA: 边叠衣服边「味覚トゥッ～」也太萌了[发嗲]'
```

可见里面主要信息为中文，其余还有一些字符、@、英文，特别要注意的是，这是微博数据，也就是说[]里的内容并不是真正的文本，而是表情，如果没有注意这个，那提取关键词的时候这些表情就会变成最多的词，因为表情才是用户发的最多的，什么[哈哈]、[哭]什么的。

值得注意的是，除了这些之外，在观察了text文本数据内容，并且在后面提取了关键词后我们会发现，“转发微博”、“微博”、“哈哈”、“啊”这些词出现的频率很高，但是并没有实际含义，而且“哈哈”和“啊”这类词的长度不定，可以是任意个“哈”和“啊”，所以不能在停用词里去掉。所以根据这些，我们来清洗text文本数据。

```
df.text.replace(r'\[.*?\]', '', regex=True, inplace=True)
df.text.replace(r'转发微博', '', regex=True, inplace=True)
df.text.replace(r'轉發微博', '', regex=True, inplace=True)
df.text.replace(r'微博', '', regex=True, inplace=True)
df.text.replace(r'转发', '', regex=True, inplace=True)
df.text.replace(r'哈哈*', '', regex=True, inplace=True)
df.text.replace(r'啊*', '', regex=True, inplace=True)
df['text'] = df['text'].str.replace(r'^[\u4e00-\u9fa5]', '')
df
```

	求一切順利
	想要全都想要
	吐槽点太多竟然没被吐槽而且竟然没有人吐槽他
	係時候迫害下大家
	他又抽了

得到的结果里只包含中文。但是有些行已经为空了，防止对我们后续造成影响，我们将这些空行去掉。首先填充NaN：

```
df.text=df.text.astype(str)
df.replace(to_replace=r'^\s*$',value=np.nan,regex=True,inplace=True)
df
```

	text
0	NaN
1	NaN
2	求一切順利
3	想要全都想要
4	NaN
5	吐槽点太多竟然没被吐槽而且竟然没有人吐槽他

然后去掉带NaN的行并将索引重排：

```
df.dropna(subset=['text'],axis=0,how='any',inplace=True)
df.reset_index(drop=True,inplace=True)
df
```

	mid	retweeted_status_mid	uid	retweeted_uid	source	image
0	mH44qG6iUm	mH44qL9LlF	uK3RXUJ0V	NaN	新浪微博	0 求一切顺利
1	mZmwFTOdVX	mcyE5GR7GJ	uK3RXUJ0V	NaN	新浪微博	0 想要全都想要
2	mnzrsoGWNN	mNfGcUeZbK	uK3RXUJ0V	NaN	新浪微博	0 吐槽点太多竟然没被吐槽而且竟然没有
3	m2rVkbmLsg	m7nJhJ3W6z	uK3RXUJ0V	NaN	新浪微博	0 係時候迫害下大家
4	mNfG6Xsbs5	mex2cwWppM	uK3RXUJ0V	NaN	新浪微博	0 他又抽了
5	mu9iEJwbEt	mPoMqa8zoK	uK3RXUJ0V	NaN	新浪微博	0 太萌了边叠衣服边味觉也太萌了

到这里为止数据就算是清理好了。

三、用户画像

先把user_df中的大V的性别和省份筛选出来。

```
big_v=pd.DataFrame()
big_v['province']=user_df.province[user_df.verified==True]
big_v['gender']=user_df.gender[user_df.verified==True]
big_v
```

	province	gender
13	11	m
16	11	m
24	44	m
41	52	m
46	11	m

查看性别分布：

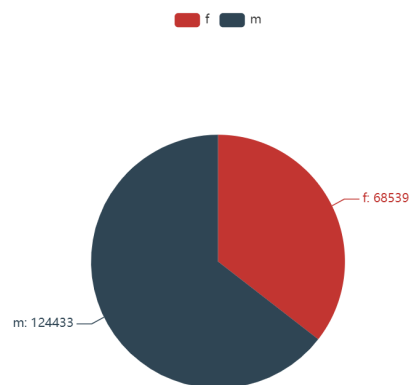
```
#y_data=list(big_v['gender']).count()
gender=big_v['gender'].value_counts()
gender
```

```
m      124433
f       68539
Name: gender, dtype: int64
```

用pyecharts画出微博大V性别分布图:

```
x_data=list(gender.index)
y_data=list(gender)
data_pair=list(z for z in zip(x_data,y_data))
data_pair.sort(key=lambda x:x[1])
pie=Pie()
pie.add('',data_pair=data_pair,radius='50%',center=
['50%','50%'])
pie.set_global_opts(title_opts=opts.TitleOpts(title='微博大
V男女比'),
                    legend_opts=opts.LegendOpts(is_show=True))
pie.set_series_opts(label_opts=opts.LabelOpts(formatter="
{b}: {c}"))
#pie.render('1.html')
pie.render_notebook()
#因为pyecharts渲染图片是从远处抓取的,所以有可能显示不出来,我就直接
把图片贴在下面了
#也可以打开html看
```

微博大V男女比



可见微博大V中男性要比女性多一点, 男性有124433个, 女性有68539个。

接下来看省份分布, 这里的省份只有代码, 所以我们要先在网上找一个省份代码对照表province.txt。内容如下:

1	code	city
2	11	北京
3	12	天津
4	13	河北
5	14	山西
6	15	内蒙古
7	21	辽宁
8	22	吉林
9	23	黑龙江
10	31	上海
11	32	江苏
12	33	浙江
13	34	安徽
14	35	福建
15	36	江西
16	37	山东
17	41	河南
18	42	湖北
19	43	湖南
20	44	广东
21	45	广西
22	46	海南

然后将表big_v中的省份代码换成真正的省份名称。

```
province=pd.read_csv('province.txt',sep='\t',skiprows=1,na
mes=['code','province'])
province1=province.set_index('code')
province1
```

	province
code	
11	北京
12	天津
13	河北
14	山西
15	内蒙古

```
code=list(province1.index)
def transfer(x):
    y=province1.loc[x]
    return y
big_v['place']=big_v.province[big_v['province'].isin(code)
].apply(transfer)
big_v
```

	province	gender	place
13	11	m	北京
16	11	m	北京
24	44	m	广东
41	52	m	贵州
46	11	m	北京
54	35	m	福建

查看省份分布：

```
place=big_v.place.value_counts()  
place
```

北京	60173
广东	20308
上海	18293
浙江	9740
江苏	8488
香港	6364
山东	5827
福建	5317
四川	4920
湖北	4259
辽宁	4138
湖南	4040
河南	4025

用pyecharts画出省份地图分布:

```
x_place=place.index
y_place=place.values
y_place=y_place.tolist()
x_place=x_place.tolist()
data_pair1=list(z for z in zip(x_place,y_place))
pieces = [
    {'min': 60000, 'color': '#540d0d'},
    {'max': 59999, 'min': 10000, 'color': '#9c1414'},
    {'max': 9999, 'min': 5000, 'color': '#d92727'},
    {'max': 4999, 'min': 2000, 'color': '#ed3232'},
    {'max': 1999, 'min': 1000, 'color': '#f27777'},
    {'max': 999, 'min': 100, 'color': '#f7adad'},
    {'max': 0, 'color': '#f7e4e4'},
]
m=Map()
m.add("大v数量",data_pair1,'china')
#系列配置项,可配置图元样式、文字样式、标签样式、点线样式等
m.set_series_opts(label_opts=opts.LabelOpts(font_size=12),
                  is_show=False)
#全局配置项,可配置标题、动画、坐标轴、图例等
m.set_global_opts(title_opts=opts.TitleOpts(title='全国各地
大v数量',pos_top=True),

legend_opts=opts.LegendOpts(is_show=False),
```

```

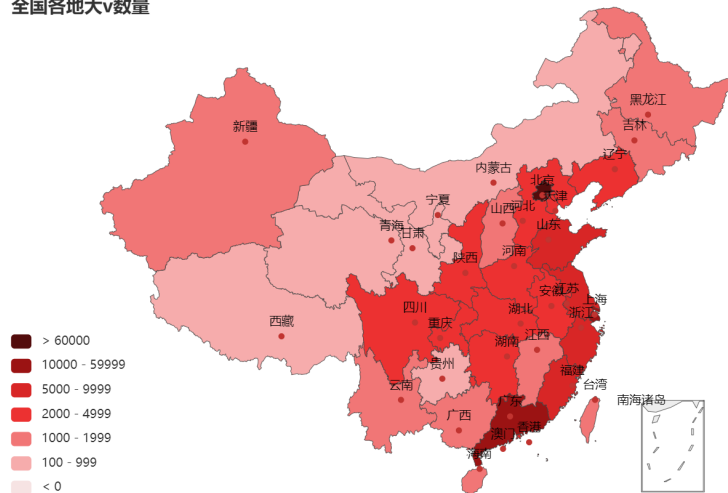
visualmap_opts=opts.VisualMapOpts(pieces=pieces,

is_piecewise=True,    #是否为分段型

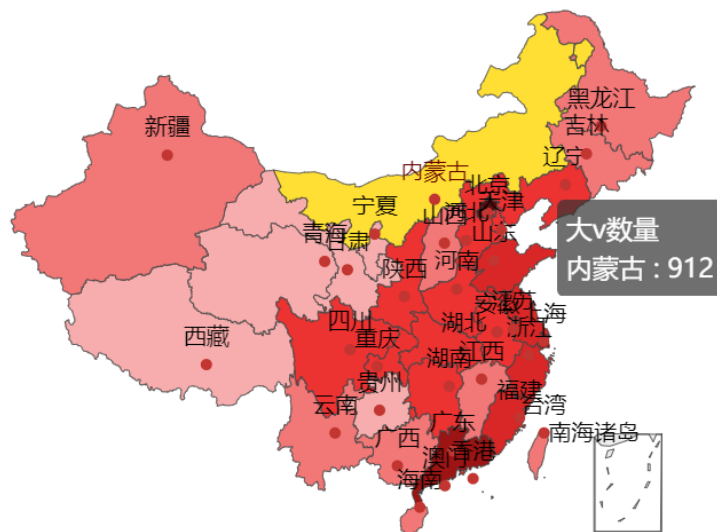
is_show=True))        #是否显示视觉映射配置
m.render_notebook()

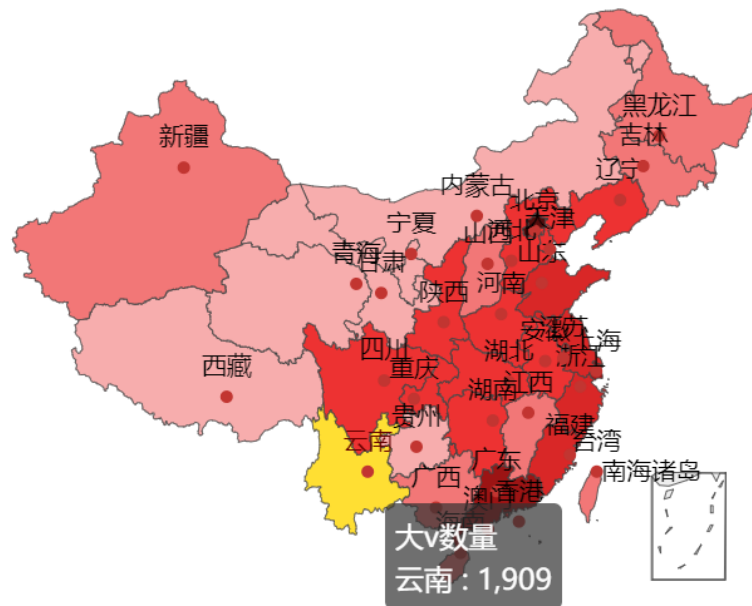
```

全国各地大v数量



图像是交互式的，可以把鼠标移到省份上查看当前省份的大V数量：





从图中可看出大V主要集中在东部地区，这应该是因为东部地区经济更加发达，资源更多，网络条件什么的也更好，相比之下成为大V更容易的缘故。

接下来看看微博大V最常使用的客户端。

这个信息在df里，所以我们将user_df中的大V的uid弄出来：

```
v_uid=list(user_df['uid'][user_df['verified']==True])
```

内容长度超过1000行，保存时将截断

```
[ 'uTZVIO0X3 ',
  'uKB5YTJX3 ',
  'uTHYACFXA ',
  'uSJMRZGMT ',
  'uRU2TSMB ',
  'u0ABQZOMR ',
  'u3FTSA5Y2 ',
  'uBIDWFZYL ',
  'uMLL4EWY3 ',
  'uKPKOBVBN ',
  'uEMBY0ZVK ',
```

然后筛选出大V的客户端来源：

```
source_df=pd.DataFrame(df.loc[df.uid.isin(v_uid)].source)
source=source_df.source.value_counts()[:10]
source
```

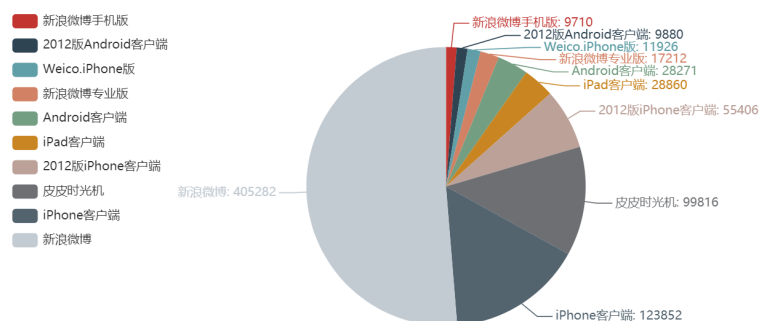
新浪微博	405282
iPhone客户端	123852
皮皮时光机	99816
2012版iPhone客户端	55406
iPad客户端	28860
Android客户端	28271
新浪微博专业版	17212
Weico.iPhone版	11926
2012版Android客户端	9880
新浪微博手机版	9710

Name: source, dtype: int64

还是用pyecharts画出饼图:

```
x_data_s=list(source.index)
y_data_s=list(source)
data_pair_s=list(z for z in zip(x_data_s,y_data_s))
data_pair_s.sort(key=lambda x:x[1])
pie=Pie()
pie.add('',data_pair=data_pair_s,radius='55%',center=
['50%','50%'])
pie.set_global_opts(title_opts=opts.TitleOpts(title='微博大
V客户端程序前十'),
legend_opts=opts.LegendOpts(orient="vertical",
pos_top="15%", pos_left="2%"))
pie.set_series_opts(label_opts=opts.LabelOpts(formatter="
{b}: {c}"))
pie.render_notebook()
```

微博大V客户端程序前十



图中可以看出用新浪微博客户端的大V最多，有405282个，其次是iPhone客户端和皮皮时光机，分别有123852和99816个。

然后对大V发布微博进行关键词提取作为大V的用户标签，分为全部大V、男性大V和女性大V。

首先对df中的text进行jieba分词：

```
def cut_words(a):
    seg=jieba.cut(a)
    return ' '.join(seg)
df['seg_words']=df['text'].apply(cut_words)
df.seg_words
```

```
求 一切 順利
想要 全都 想要
吐槽 点太多 竟然 没 被 吐 槽 而且 竟然 没有 人吐槽 他
係 時候 迫害 下 大家
他又 抽 了
太萌 了 边 叠 衣服 边 味 覺 也 太 萌 了
霸气 嘍 好 霸气 明天 真的 满满
利达 从头到尾 都 很 紧张 的 样子 让 看着 的 我 都 担心 起来 除了 要 好好 的...
美帅
年末 真 好看
這我 看 不下 十次 了 覺得 比 逃走 更好 笑
嘍真 纯情 嘍 好 闷骚 你 是 宅 男 吗
沒事 吧
最喜歡 這張 了
喂 一直 都 很 亞撒西 小字 與 相葉君 一起 去 看 了 松本 潤 的 舞台 劇 荒野 ...
```

把男性大V和女性大V的uid分别取出：

```
m_v_uid=list(user_df['uid'][user_df['verified']==True]
[user_df['gender']=='m'])
f_v_uid=list(user_df['uid'][user_df['verified']==True]
[user_df['gender']=='f'])
```

然后分别把所有大V、男性大V、女性大V的微博内容筛选出来：

```
m_text=list(df.loc[df.uid.isin(m_v_uid)].seg_words)
f_text=list(df.loc[df.uid.isin(f_v_uid)].seg_words)
v_text=list(df.loc[df.uid.isin(v_uid)].seg_words)
```

开始提取前100个关键词：

```
m = jieba.analyse.extract_tags(m_text,topK=100)
f = jieba.analyse.extract_tags(f_text,topK=100)
v = jieba.analyse.extract_tags(v_text,topK=100)
```

结果：

男性大V：

```
'分享 中国 新年 朋友 年月日 喜欢 春运 关注 谢谢 生活 希望 人生 北京 时间 幸福 老师 支持 工作 活动 感谢 手机 世界 新浪 孩子 视频 真的 明天 美国
晚安 快乐 童鞋 回家 加油 推荐 新闻 早安 电影 照片 网络 元旦 媒体 博文 地址 期待 图片 成功 一种 男人 恭喜 发现 感觉 公司 网友 一年 现场 香港 报道
企业 直播 今日 不错 节目 选择 春节 今晚 社会 健康 精彩 文化 开心 网站 祝福 粉丝 女人 上海 设计 东西 小时 私信 音乐 国家 努力 有人 电话 服务 广告
生命 城市 同学 游戏 学习 爱情 日本 机会 地方 新年快乐 警方 购票 市场 苹果'
```

女性大V：

所有大V:

词云可视化:

男性大V:



 image-20200510222919320

所有大V:

 image-20200510223213261

四、热点话题分析

首先我们来提取排名前十的热点话题。

因为话题并不等同于关键词，而更像是一些关键词的组合，所以像上面那样直接提取关键词作为话题似乎并不科学。所以我们这里用LDA（隐含狄利克雷分布）来进行文本主题抽取。

LDA 在主题模型中占有非常重要的地位，常用来文本分类。LDA由Blei, David M.、Ng, Andrew Y.、Jordan于2003年提出，用来推测文档的主题分布。它可以将文档集中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取它们的主题分布后，便可以根据主题分布进行主题聚类或文本分类。

机器学习的模型分为两种，一种是基于策略，即不能给出明确的数据分布的，一种是基于模型，可以给出分布的形式，但是超参数不知道。kmeans，dbscan是基于性能和密度的，基于策略寻找最优聚类方案，而PLSA和LDA是基于多项式分布和狄利克雷分布的，基于参数迭代寻找最优聚类方案的。

先对文本进行jieba分词：

```
def cut_words(a):  
    seg=jieba.cut(a)  
    return ' '.join(seg)  
df['seg_words']=df['text'].apply(cut_words)  
df
```

seg_words

求一切順利

想要全都想要

吐槽点太多竟然没被吐槽而且竟然没有人吐槽他

係時候迫害下大家

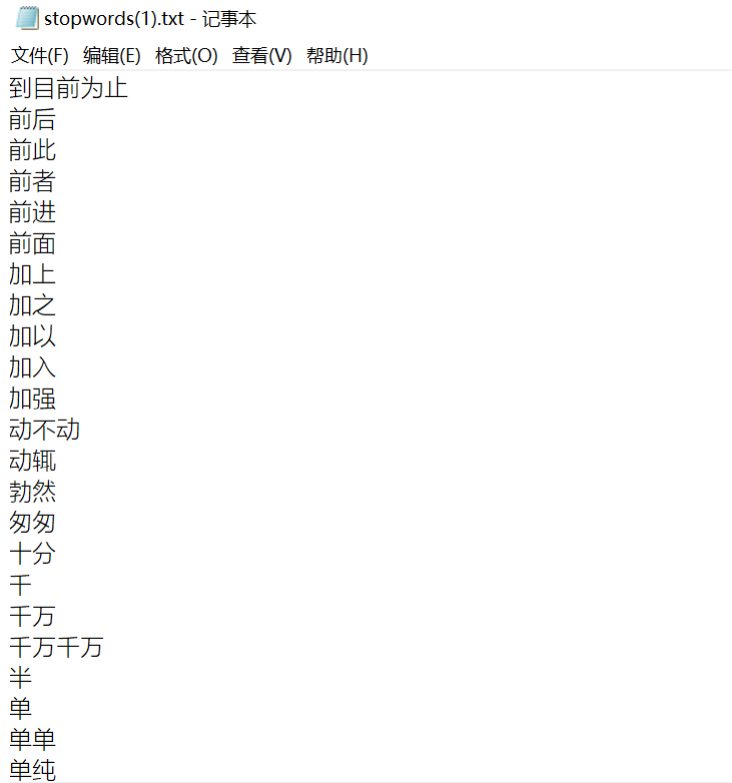
他又抽

单词之间都被空格区别开来，接下来对文本做向量化。

导入包：

```
#文本向量化
from sklearn.feature_extraction.text import
TfidfVectorizer,CountVectorizer
```

导入停用词（停用词就是一些经常在文中出现但是没有太重要意义的词），这里用的是网上找的停用词表：



```
stopwords(1).txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
到目前为止
前后
前此
前者
前进
前面
加上
加之
加以
加入
加强
动不动
动辄
勃然
匆匆
十分
千
千万
千万千万
半
单
单单
单纯
```

```
stopword="stopwords.txt"
with open(stopword,'rb') as f:
    stopwords=f.read().decode('utf-8') #停用词提取
stoplist=stopwords.splitlines()
```

如果处理太多词汇的话时间太长，于是这里我就只从文本中提取1000个最重要的特征关键词，并进行向量转换。

```
n_features=1000
tf_vectorizer=TfidfVectorizer(
    max_features=n_features,
    stop_words=stoplist,
    max_df=0.5,
    min_df=10
)
tf=tf_vectorizer.fit_transform(df.seg_words)
tf.shape
```

(3777777, 1000)

提取的向量矩阵大小为（3777777,1000）。

导入LDA包：

```
from sklearn.decomposition import
LatentDirichletAllocation
```

LDA需要人为设定主题数量，这里我选择30个，然后再取前10个主题。

```
n_topics=30
lda=LatentDirichletAllocation(n_topics=n_topics,
                              #learning_method='batch',
                              learning_method='online',
                              max_iter=20,
                              learning_offset=50.,
                              random_state=0,
                              batch_size=128,
                              verbose=1)

lda.fit(tf)
```

主题并没有确定的名称，而是用一系列关键词相结合形成的。模型fit后，需要定义每个主题输出多少个关键词，这里暂定输出前20个关键词：

```
def print_topic(model,feature_names,n_top_words):
    for idx,topic in enumerate(model.components_):
        if(idx<10):
            print("Topic{}".format(idx))
            print(" ".join([feature_names[i] for i in
                             topic.argsort()[::-n_top_words-1:-1]]))
        else:
            break
    print()

n_top_words=20
tf_feature_names=tf_vectorizer.get_feature_names()
print_topic(lda,tf_feature_names,n_top_words)
```

输出结果：

```
Topic0
关注 电影 快乐 两个 经典 一个 精彩 评论 懂得 明星 直播 事件 调查 专业 放在 性感 一段 最美 一部 一名
Topic1
上海 今日 我要 微笑 节目 真实 家里 话题 答应 数据 神奇 放假 估计 礼物 眼泪 有奖 覺得 全世界 失望 悲伤
Topic2
爱情 悄悄 国家 冬天 年月日 早安 搭配 心理 女性 年轻 趣味 减肥 理解 不行 生日快乐 身上 送给 分手 蛋糕 其實
Topic3
活动 祝福 心愿 一年 感谢 一家 机会 参加 游戏 粉丝 享受 一张 好友 即可 结束 提供 休息 元旦 博文 目标
Topic4
健康 晚安 早就 童鞋 不到 睡觉 最终 越来越 考试 收藏 日子 一份 报道 学校 速度 尼玛 收到 平安 瞬间 回答
Topic5
香港 美国 全球 喜欢 南京 什麼 下次 时候 虽然 标准 时光 人才 時間 合作 已經 那麼 公开 没有 干嘛 武汉
Topic6
女人 人生 成功 定义 设计 适合 不用 放弃 神马 寻找 坤哥 用户 记住 路上 心灵 烦恼 风景 国内 欢乐 便宜
Topic7
地址 投票 参与 领导 价值 春节 一辈子 痛苦 霸气 小心 心中 提高 十大 投给 发起 表态 选项 一只 自我 周末
```

Topic8

回家 手机 恭喜 名字 全国 品牌 免费 刚刚 宝贝 家人 有个 勋章 所有人 链接 杭州 小孩 介绍 口味 带上 干净

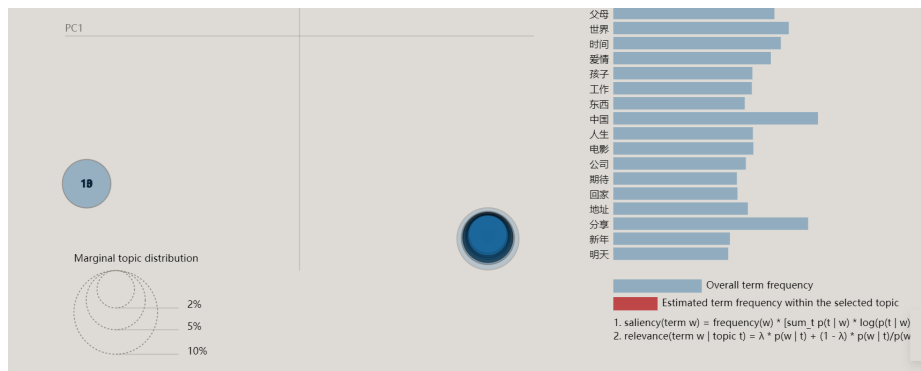
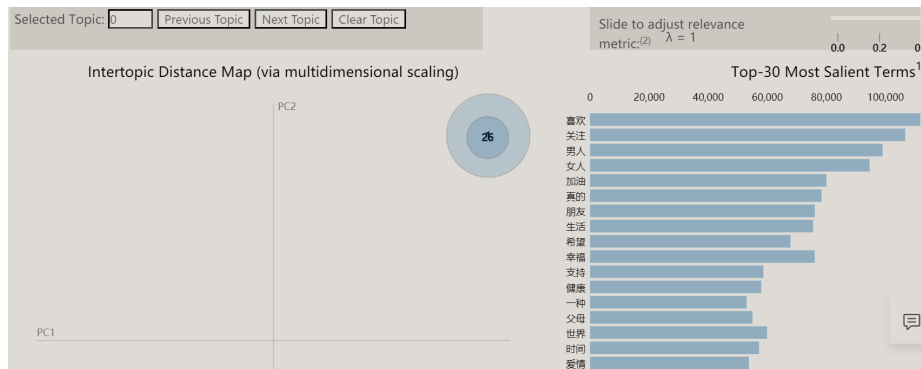
Topic9

珍惜 离开 效果 這個 阳光 精神 保护 大学 更好 伤害 三个 坚强 青春 长途旅行 還是 系列 解决 甜蜜 二次元 起来

可以大概看出每个主题大概是什么。比如第一个大概是关于电影明星的，第四个大概是抽奖、祝福之类的，第十个大概是关于人生鸡汤之类的。

这里可以用pyLDAvis来可视化：

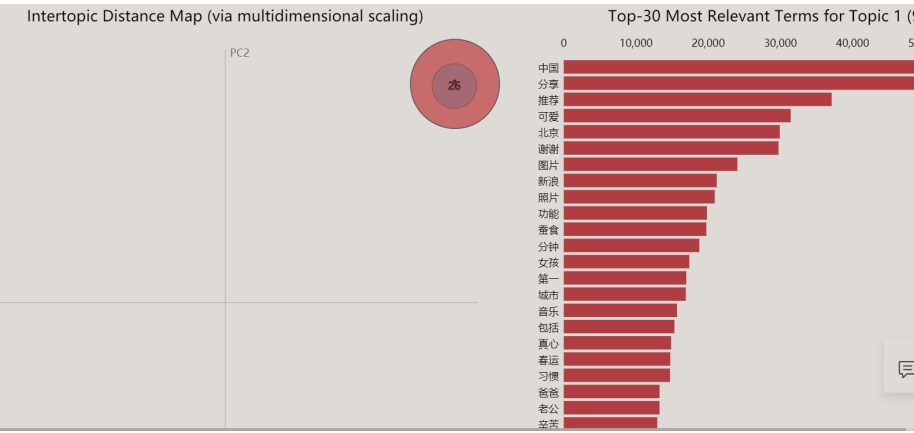
```
import pyLDAvis
import pyLDAvis.sklearn
pyLDAvis.enable_notebook()
pyLDAvis.sklearn.prepare(lda,tf,tf_vectorizer)
```



在图的左侧，是用圆圈代表不同的主题，圆圈的大小代表每个主题分别包含文章的数量。可以看到30个主题大概聚成了三堆，聚成一堆的代表他们主题比较相似。

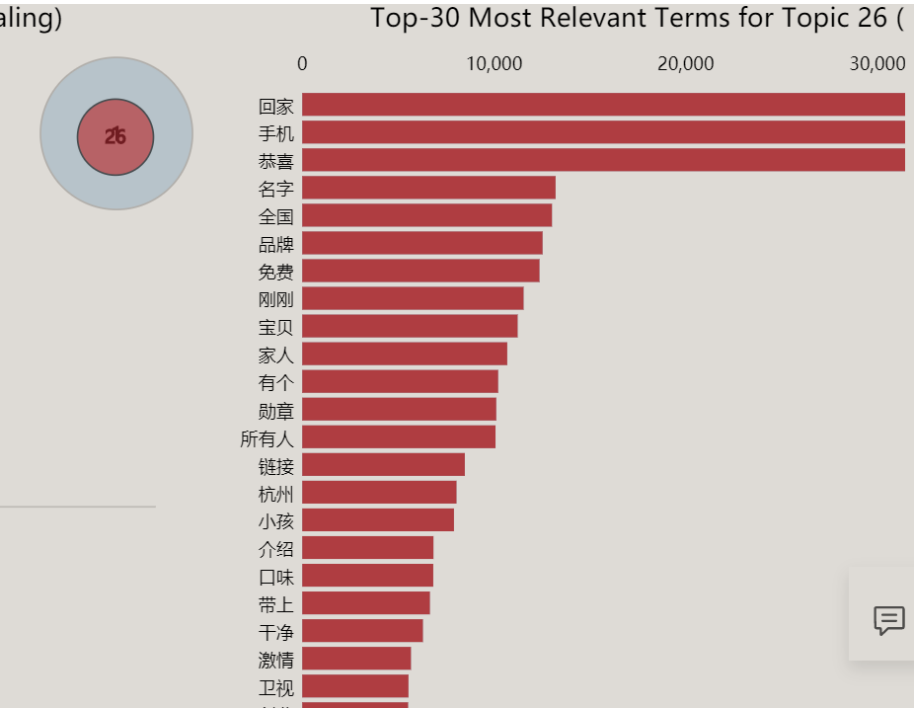
当鼠标没有指向任何主题的时候，右侧的30个关键词代表全部文本中提取到的30个最重要关键词。大概是“喜欢、关注、男人、女人、加油”这些词。

如果把鼠标放到某个主题（这里是主题一）下：



右侧的关键词列表就会变化，红色展示了每个关键词在当前主题下的频率。

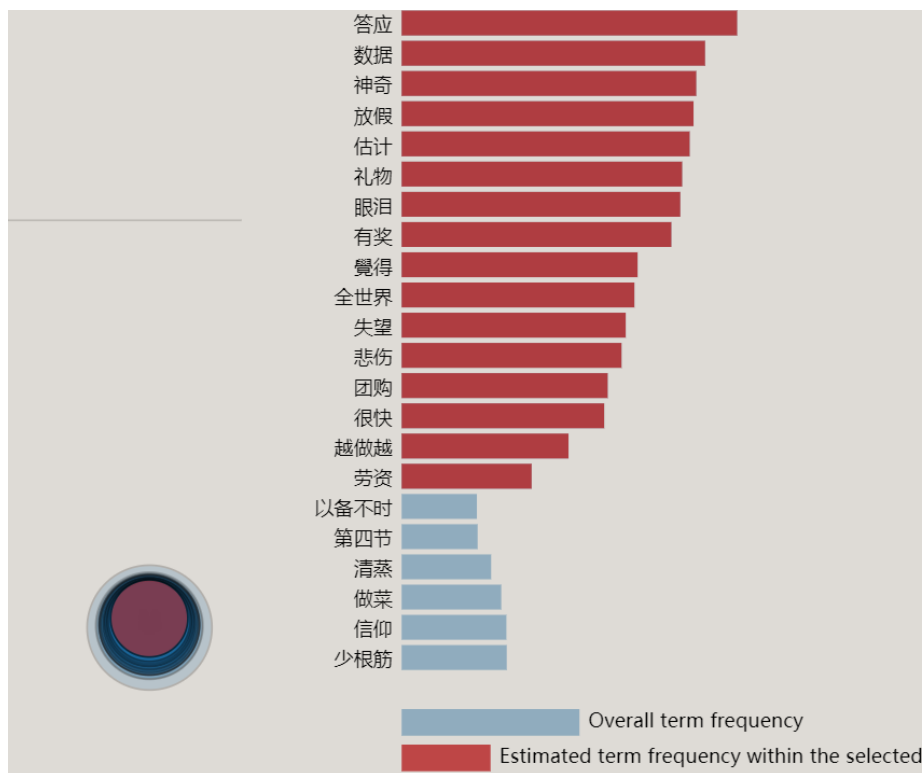
这里主题主要被分成了三个区域，对于右上角的区域，可以看到大概是关于地区、国家的。



对于左下角的区域，可以大概看出是关于政治、技术什么的：



左下角的区域聚集最多，大概是关于生活、星座什么的的东西：



因为只迭代了20次，所以效果不是太好，但是大概还是可以看出的。

接下来我们分析不同时间段的热点话题，看下热点话题的演化过程。

时间数据在df的created_at字段里，先来看一下数据：

```
0    2012-01-03 01:15:36
1    2012-01-03 01:12:55
2    2012-01-03 01:09:54
3    2012-01-03 01:08:45
4    2012-01-03 00:54:07
5    2012-01-03 00:38:10
6    2012-01-02 22:58:52
7    2012-01-02 22:51:06
8    2012-01-02 20:46:42
9    2012-01-02 20:43:18
Name: created_at, dtype: object
```

这些数据是object，不好操作，所以需要把它转化成时间序列。

```
#将created_at转成时间序列
time_data=pd.to_datetime(df['created_at'],format='%Y-%m-%d')
df['time']=time_data
```

然后把它作为索引并排序：


```

stop_words=stoplist,
max_df=0.5,
min_df=10)
tf_vectorizer_2=CountVectorizer(strip_accents='unicode',
max_features=n_features,
stop_words=stoplist,
max_df=0.5,
min_df=10)
tf_vectorizer_3=CountVectorizer(strip_accents='unicode',
max_features=n_features,
stop_words=stoplist,
max_df=0.5,
min_df=10)
tf_1=tf_vectorizer_1.fit_transform(date_1)
tf_2=tf_vectorizer_2.fit_transform(date_2)
tf_3=tf_vectorizer_3.fit_transform(date_3)

```

进行LDA主题抽取，这里只选取5个主题：

```

n_topics=5
lda_1=LatentDirichletAllocation(n_topics=n_topics,
                                #learning_method='batch',
                                learning_method='online',
                                max_iter=20,
                                learning_offset=50.,
                                random_state=0,
                                batch_size=128,
                                verbose=1)
lda_2=LatentDirichletAllocation(n_topics=n_topics,
                                #learning_method='batch',
                                learning_method='online',
                                max_iter=20,
                                learning_offset=50.,
                                random_state=0,
                                batch_size=128,
                                verbose=1)
lda_3=LatentDirichletAllocation(n_topics=n_topics,
                                #learning_method='batch',
                                learning_method='online',
                                max_iter=20,
                                learning_offset=50.,
                                random_state=0,
                                batch_size=128,
                                verbose=1)

```

第一阶段的热点话题：

Topic0
希望 人生 感觉 发现 快乐 回家 一年 生命 美丽 事情 老师 咖啡 开心 二次元 感谢 不想 心情 元旦 登场 告诉
Topic1
真的 支持 活动 地址 明天 北京 电影 香港 不错 可爱 期待 加油 上海 好好 广州 成功 努力 深圳 调查 投票
Topic2
喜欢 中国 关注 改变 一种 谢谢 想要 今晚 永远 感受 分享 环境 美国 图片 简单 公司 所有人 收藏 这是 心态
Topic3
生活 朋友 吴奇隆 世界 好多 幸福 開始 還有 穿穿 成果 花絮 起來 雜誌 熊貓 不斷 族人 脫脫 戰神 愛特 时间
Topic4
男人 女人 新年 孩子 分享 东西 推荐 手机 终于 有人 坤哥 视频 一点 时尚 晚上 劳资 分钟 音乐 值得 新浪

第二阶段的热点话题：

Topic0
命运 真的 木有 用千本 本来 感到 负责 有空 惊喜 一段 生日 双子 熟悉 恋爱 原来 女孩 事件 放入 更好 选择
Topic1
天蝎座 小编 市民 一生 父亲 人气 妈妈 尊重 善良 国际 友情 我們 微笑 美女 這個 世界 做人 美丽 事业 担心
Topic2
世上 收获 春节 口味 女儿 东莞 什麼 感动 迪拜 不斷 形象 怎麼 创新 环境 大学 领导 白羊座 面对 分秒 参与
Topic3
公司 思考 几天 刷屏 推荐 明星 不错 努力 刺激 香港 方向 名字 电影 困难 一家 女子 手机 明天 管理 年月日
Topic4
电话 严肃 明明 深圳 全国 流行 图片 老师 新鲜 感覺 功能 今日 某人 记得 好玩 特色 生命 太阳 神马 近日

第三阶段的热点话题：

Topic0
父母 健康 世界 祝福 心愿 人生 儿女 孩子 电影 明天 谢谢 好好 简单 我們 时尚 晚安 真的 天下 女孩 记得
Topic1
喜欢 分享 关注 推荐 回家 不错 北京 手机 新年 图片 开心 终于 心情 今晚 视频 新浪 分钟 妈妈 赶紧 童鞋
Topic2
幸福 朋友 生活 希望 时间 一种 快乐 香港 永远 发现 东西 感觉 老师 努力 两个 选择 成功 地方 美国 生命
Topic3
女人 加油 男人 真的 公司 支持 柯达 地址 定义 蚕食 期待 活动 一家 可爱 胶片 数码 事情 市场 好消息 早就
Topic4
中国 星座 爱情 工作 恭喜 结婚 有人 旅行 建议 春运 包括 婚姻 不想 最终 高智商 那种 一年 天蝎座 公认 狮子座

最后来总结一下热点话题形成的特点和关键因素。

首先当然是微博大V发布和转发的消息会更容易成为热点话题，而从大V的用户画像中可以看出来自北京、上海、广东等发达地区的大V数量更多，他们引起的热点话题也就多。

其次是一条微博的转发量。转发量越多，自然话题变成热点话题的概率也就更大。

还有就是各种重大节日和事件的发生。比如像是情人节，恋爱、感情就会变成热点话题，而如果是奥运会开幕这种事件，运动员就会变成热点话题。热点话题是随着时间和事件的发生决定的。

而关于工作、生活、学习之类的事情，其实一直都是热点话题，一直被人们所讨论。

同时，男性和女性各自的热点话题也会有所差异。

以上就是我的微博大V数据分析报告。详细代码见ipynb文档。