一、Python selenium爬取微博内容

（一）编程环境

操作系统：Win 10

编程语言：Python 3.6

（二）安装selenium

这里使用selenium实现。

如果没有安装过python的selenium库，则安装命令如下

```

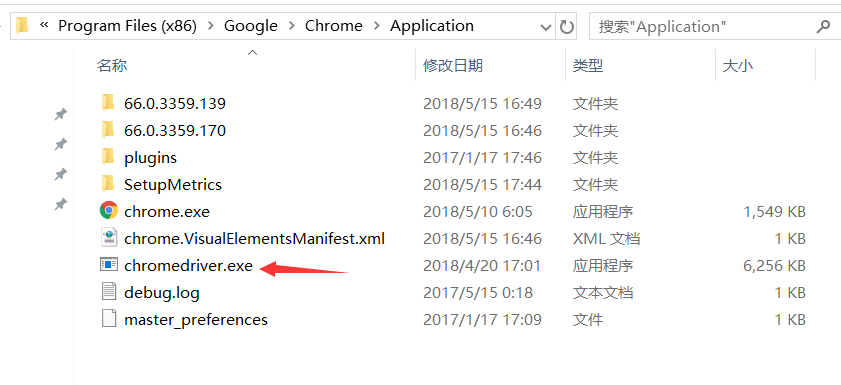
pip install selenium

```

（三）下载ChromeDriver

因为selenium要用到浏览器的驱动，这里我用的是Google Chrome浏览器，所以要先下载ChromeDriver.exe并放到C:\Program Files (x86)\Google\Chrome\Application\目录下。

注意，放到别的目录下也行，只要在代码里填上正确的路径即可。



（四）登录微博

通常而言，m站的网页结构比pc站要简单的多，咱们可以从m站入手。微博m站登录界面的网址是 <https://passport.weibo.cn/signin/login>

在Chrome浏览器中打开此地址，并右键点击界面的任何位置-->查看网页源代码，发现邮箱/手机号框的id为loginName，密码输入框的id为loginPassword，登录按纽的id为loginAction。

```

from selenium import webdriver

import time

#全局变量

driver = webdriver.Chrome("C:\Program Files (x86)\Google\Chrome\Application\chromedriver.exe")

def loginWeibo(username, password):

driver.get('https://passport.weibo.cn/signin/login')

time.sleep(3)

driver.find\_element\_by\_id("loginName").send\_keys(username)

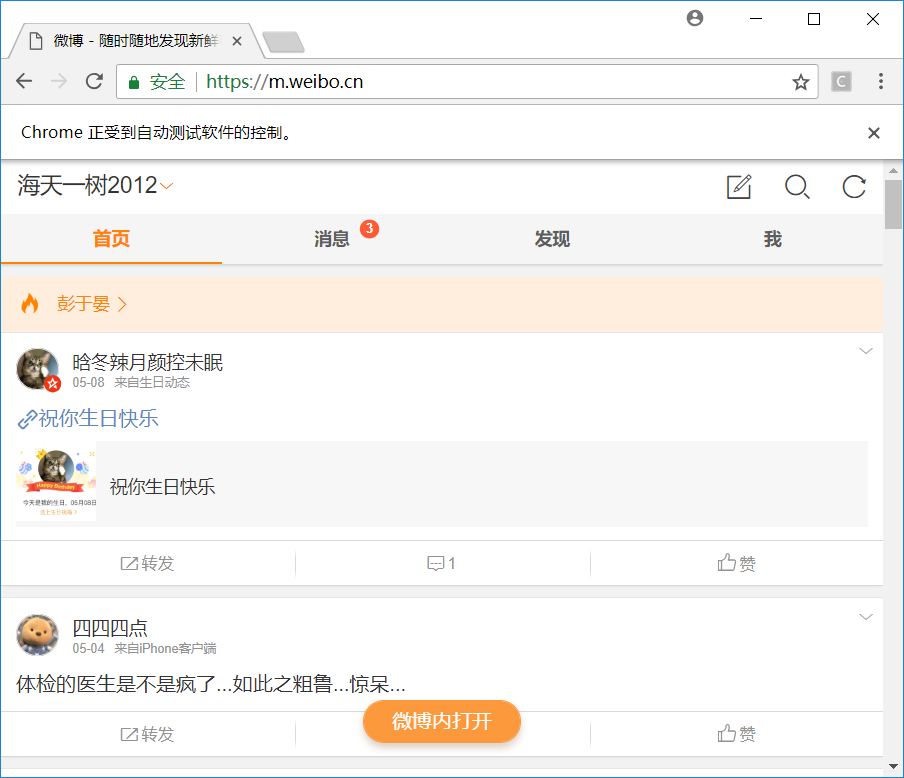
driver.find\_element\_by\_id("loginPassword").send\_keys(password)

driver.find\_element\_by\_id("loginAction").click()

#driver.close()

```

执行后可以看到自动登录的过程及登录成功的界面



（五）爬取微博内容

从微博爬取内容有两种方式：

（1）申请成为新浪开发者并调用微博API

（2）使用爬虫程序

因为微博API有好多限制，比如只能获取某用户的最近的10条微博内容，而不能获取全部历史微博内容。

这里咱们采用爬虫方式。

程序如下：

```

from selenium import webdriver

import time

import re

#全局变量

driver = webdriver.Chrome("C:\Program Files (x86)\Google\Chrome\Application\chromedriver.exe")

def loginWeibo(username, password):

driver.get('https://passport.weibo.cn/signin/login')

time.sleep(3)

driver.find\_element\_by\_id("loginName").send\_keys(username)

driver.find\_element\_by\_id("loginPassword").send\_keys(password)

driver.find\_element\_by\_id("loginAction").click()

#这里只是看一下cookie内容，下面不会用到这个cookie值，因为driver会把cookie自动带过去

cookies = driver.get\_cookies()

cookie\_list = []

for dict in cookies:

cookie = dict['name'] + '=' + dict['value']

cookie\_list.append(cookie)

cookie = ';'.join(cookie\_list)

print (cookie)

#driver.close()

def visitUserInfo(userId):

driver.get('http://weibo.cn/' + userId)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print('用户资料')

# 1.用户id

print('用户id:' + userId)

# 2.用户昵称

strName = driver.find\_element\_by\_xpath("//div[@class='ut']")

strlist = strName.text.split(' ')

nickname = strlist[0]

print('昵称:' + nickname)

# 3.微博数、粉丝数、关注数

strCnt = driver.find\_element\_by\_xpath("//div[@class='tip2']")

pattern = r"\d+\.?\d\*" # 匹配数字，包含整数和小数

cntArr = re.findall(pattern, strCnt.text)

print(strCnt.text)

print("微博数：" + str(cntArr[0]))

print("关注数：" + str(cntArr[1]))

print("粉丝数：" + str(cntArr[2]))

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

# 4.将用户信息写到文件里

with open("userinfo.txt", "w", encoding = "gb18030") as file:

file.write("用户ID：" + userId + '\r\n')

file.write("昵称：" + nickname + '\r\n')

file.write("微博数：" + str(cntArr[0]) + '\r\n')

file.write("关注数：" + str(cntArr[1]) + '\r\n')

file.write("粉丝数：" + str(cntArr[2]) + '\r\n')

def visitWeiboContent(userId):

pageList = driver.find\_element\_by\_xpath("//div[@class='pa']")

print(pageList.text)

pattern = r"\d+\d\*" # 匹配数字，只包含整数

pageArr = re.findall(pattern, pageList.text)

totalPages = pageArr[1] # 总共有多少页微博

print(totalPages)

pageNum = 1 # 第几页

numInCurPage = 1 # 当前页的第几条微博内容

curNum = 0 # 全部微博中的第几条微博

contentPath = "//div[@class='c'][{0}]"

#while(pageNum <= 3):

while(pageNum <= int(totalPages)):

try:

contentUrl = "http://weibo.cn/" + userId + "?page=" + str(pageNum)

driver.get(contentUrl)

content = driver.find\_element\_by\_xpath(contentPath.format(numInCurPage)).text

#print("\n" + content) # 微博内容，包含原创和转发

if "设置:皮肤.图片.条数.隐私" not in content:

numInCurPage += 1

curNum += 1

with open("weibocontent.txt", "a", encoding = "gb18030") as file:

file.write(str(curNum) + '\r\n' + content + '\r\n\r\n')

else:

pageNum += 1 # 抓取新一页的内容

numInCurPage = 1 # 每一页都是从第1条开始抓

time.sleep(20) # 要隔20秒，否则会被封

except exception as e:

print("curNum:" + curNum)

print(e)

finally:

pass

print("Load weibo content finished!")

if \_\_name\_\_ == '\_\_main\_\_':

username = '\*\*\*\*\*\*' # 输入微博账号

password = '\*\*\*\*\*\*' # 输入密码

loginWeibo(username, password) # 要先登录，否则抓取不了微博内容

time.sleep(3)

uid = 'xywyw' # “寻医问药”的个性域名

visitUserInfo(uid) # 获取用户基本信息

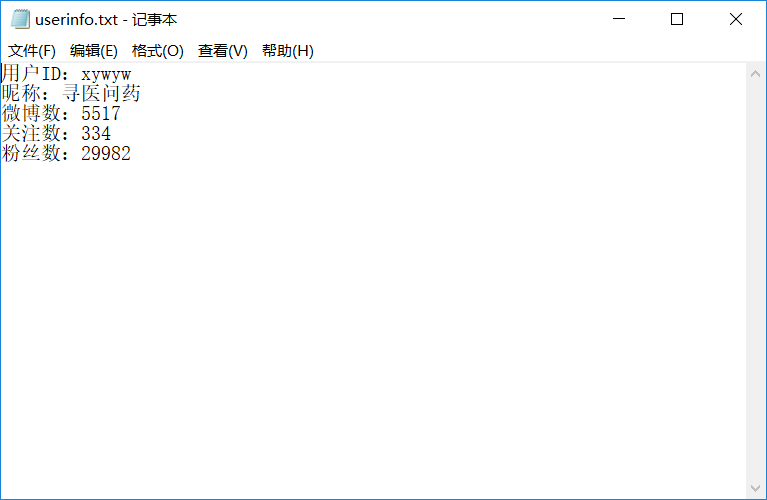
visitWeiboContent(uid) # 获取微博内容

```

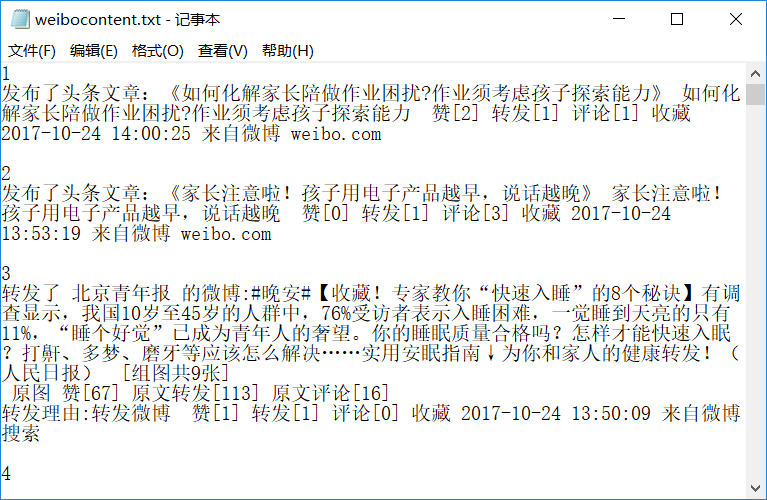
运行结果：

生成了两个文件

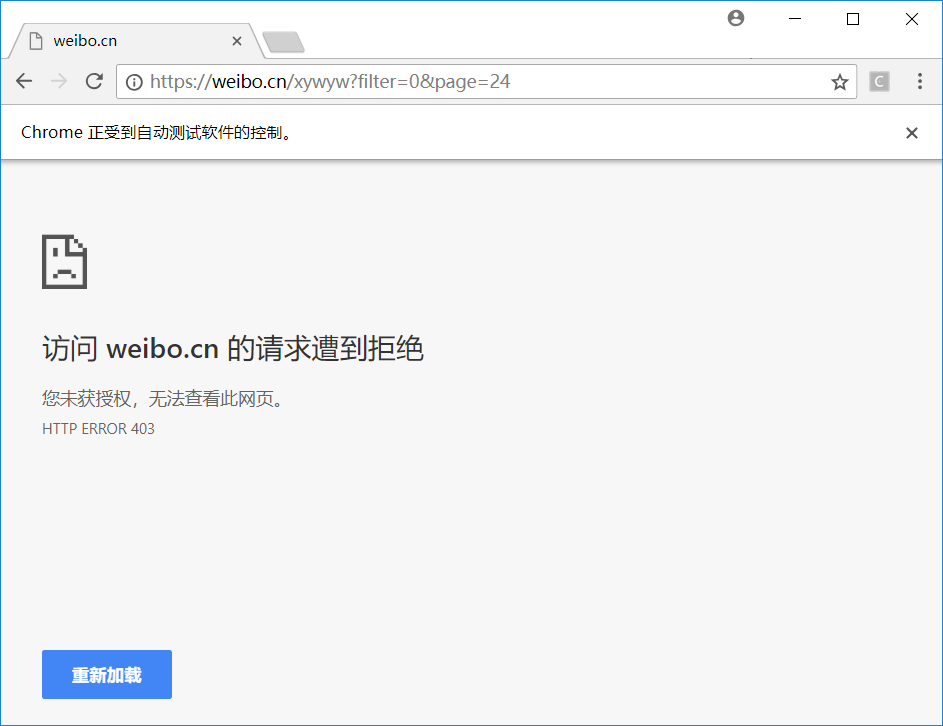
1）userinfo.txt，用来存放用户的基本资料



2）weiboconten.txt用来存放该用户的所有历史微博内容



注意，程序里每爬一页，都需要睡眠20秒。这样可以防止被封，被封会提示HTTP错误403



猜测新浪是根据cookie来封的，不是根据用户名和IP。因为403页面出现进，把浏览器的cookie清空，webo.cn就可以立即访问。如果不清空的话，等半个小时左右，也可以继续访问。

这里20秒是多次试验得到的经验值。若改成10秒，爬取200多条后会出现403错误；若改成15秒，爬取500多条后会出现403错误。

（六）微博内容处理

上面得到的微博内容，有好些无用信息，比如“赞[10]”，“发布了头条文章”等，这些内容对于后续的文本处理，没有任何作用。可以把没用的信息都处理掉：

```

fread = open('weibocontent.txt')

fwrite = open('weibocontent2.txt', 'w')

try:

for line in fread:

text = line.strip()

if("" == text):

continue

if(4 >= len(text)): #第几条微博，总共5000多条，所以不超过四位数

continue

if(-1 != text.find("转发理由")):

continue

if(-1 != text.find("原图")):

continue

pos1 = 0

if("发布了" in text or "转发了" in text):

if(-1 != text.find("：")):

pos1 = text.find("：") + 1

elif(-1 != text.find(":")):

pos1 = text.find(":") + 1

pos2 = len(text)

if(-1 != text.find("http")):

pos2 = text.find("http")

elif(-1 != text.find("全文")):

pos2 = text.find("全文")

elif (-1 != text.find("赞")):

pos2 = text.find("赞")

elif(-1 != text.find("[组图共")):

pos2 = text.find("[组图共")

content = text[pos1 : pos2]

fwrite.write(content + '\r\n')

finally:

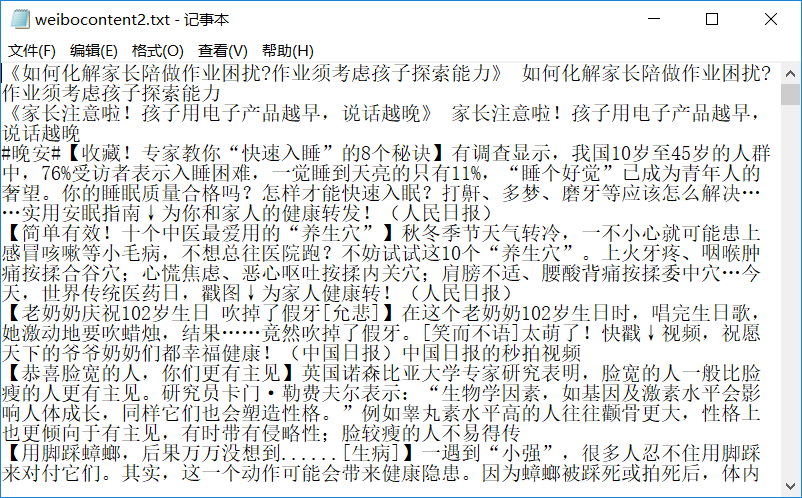
fread.close()

fwrite.close()

```

运行结果

生成weibocontent2.txt，内容为：



二、结巴分词

代码：

```

import jieba

def read\_file\_cut():

source\_file = open('weibocontent2.txt')

res\_file = open("cut\_word\_result.txt", 'w')

cnt = 0;

with open('weibocontent2.txt') as source:

content = source.read()

seglist = jieba.cut(content,cut\_all=False) #精确模式

output = ' '.join(list(seglist)) #空格拼接

with open('cut\_word\_result.txt', 'w') as target:

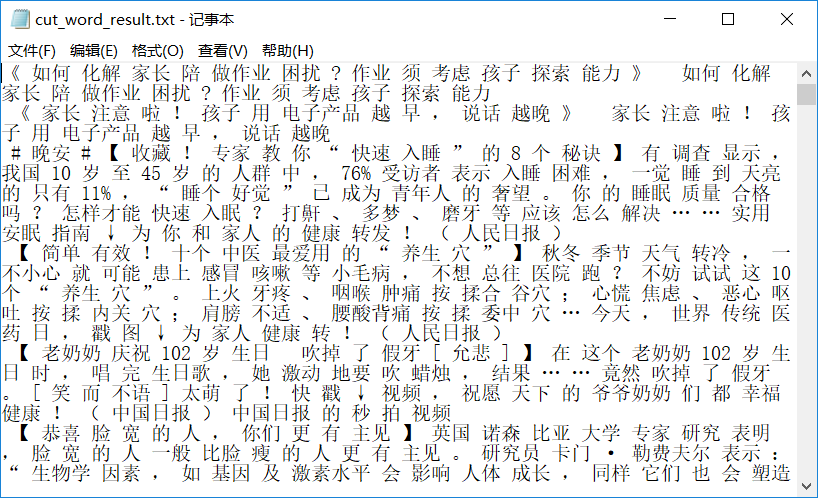
target.write(output)

if \_\_name\_\_ == '\_\_main\_\_':

read\_file\_cut()

```

运行结果：



三、使用LDA进行主题分类

代码

````

from sklearn import feature\_extraction

from sklearn.feature\_extraction.text import CountVectorizer

if \_\_name\_\_ == "\_\_main\_\_":

corpus = []

for line in open('cut\_word\_result.txt', 'r').readlines():

corpus.append(line.strip())

#print (corpus)

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(corpus)

word = vectorizer.get\_feature\_names() # 所有的特征词，即关键词

#print (word)

#print(X)

analyze = vectorizer.build\_analyzer()

weight = X.toarray()

print(weight)

import numpy as np

import lda

# 训练模型

model = lda.LDA(n\_topics = 10, n\_iter = 500, random\_state = 1)

model.fit(np.asarray(weight))

# 主题-词分布

topic\_word = model.topic\_word\_ #生成主题以及主题中词的分布

print("topic-word:\n", topic\_word)

# 计算topN关键词

topicN = 10

for i, word\_weight in enumerate(topic\_word):

#print("word\_weight:\n", word\_weight)

distIndexArr = np.argsort(word\_weight)

#print("distIndexArr:\n", distIndexArr)

topN\_index = distIndexArr[:-(topicN+1):-1]

#print("topN\_index:\n", topN\_index) # 权重最大的n个

topN\_words = np.array(word)[topN\_index]

print(u'\*Topic {}\n- {}'.format(i, ' '.join(topN\_words)))

# 绘制主题-词分布图

import matplotlib.pyplot as plt

f, ax= plt.subplots(5, 1, figsize=(7, 7), sharex=True)

for i, k in enumerate([0, 2, 4, 6, 8]): #5个主题

ax[i].stem(topic\_word[k,:1000], linefmt='b-',

markerfmt='bo', basefmt='w-')

ax[i].set\_xlim(0,1000)

ax[i].set\_ylim(0, 0.012)

ax[i].set\_ylabel("Prob")

ax[i].set\_title("topic {}".format(k))

ax[i].set\_xlabel("word")

plt.tight\_layout()

plt.show()

# 文档-主题分布

doc\_topic = model.doc\_topic\_

print("type(doc\_topic): {}".format(type(doc\_topic)))

print("shape: {}".format(doc\_topic.shape))

label = []

for i in range(10):

print(doc\_topic[i])

topic\_most\_pr = doc\_topic[i].argmax()

label.append(topic\_most\_pr)

print("doc: {} topic: {}".format(i, topic\_most\_pr))

print("前10篇文档对应的主题列表：", label) # 前10篇文档对应的主题列表

# 绘制文档-主题分布图

import matplotlib.pyplot as plt

f, ax= plt.subplots(5, 1, figsize=(7, 7), sharex=True)

for i, k in enumerate([0, 1, 2, 3, 4]):

ax[i].stem(doc\_topic[k,:], linefmt='r-',

markerfmt='ro', basefmt='w-')

ax[i].set\_xlim(-1, 10) #x坐标

ax[i].set\_ylim(0, 1.1) #y坐标

ax[i].set\_ylabel("Probability")

ax[i].set\_title("Document {}".format(k))

ax[i].set\_xlabel("Topic")

plt.tight\_layout()

plt.show()

```

运行结果

（1）lda基本信息

INFO:lda:n\_documents: 5504

INFO:lda:vocab\_size: 25841

INFO:lda:n\_words: 190021

INFO:lda:n\_topics: 10

INFO:lda:n\_iter: 500

（2）词所占的权重

[[4.57950735e-07 4.57950735e-07 4.62530242e-05 ... 4.57950735e-07

4.57950735e-07 4.57950735e-07]

[2.69103597e-07 2.69103597e-07 2.69103597e-07 ... 2.69103597e-07

2.69103597e-07 2.69103597e-07]

[5.25419534e-07 5.25419534e-07 5.25419534e-07 ... 5.25419534e-07

5.25419534e-07 5.25419534e-07]

...

[5.71839969e-07 5.71839969e-07 5.71839969e-07 ... 5.71839969e-07

5.71839969e-07 5.71839969e-07]

[9.70680218e-04 4.85097560e-07 4.85097560e-07 ... 4.85097560e-07

4.85097560e-07 9.75046096e-05]

[1.26017433e-06 1.26017433e-06 1.26017433e-06 ... 1.26017433e-06

1.26017433e-06 1.26017433e-06]]

（3）每个主题比重最大的10个词

\*Topic 0

- 导致 研究 影响 引起 可能 疾病 发现 如果 造成 容易

\*Topic 1

- 食物 维生素 含有 可以 蔬菜 预防 作用 丰富 水果 富含

\*Topic 2

- 寻医问药 自己 我们 一个 晚安 早安 不是 就是 人生 没有

\*Topic 3

- 我们 很多 食物 健康 身体 大家 饮食 可以 注意 一些

\*Topic 4

- 女人 孩子 男人 自己 健康 工作 心理 喜欢 生活 一点

\*Topic 5

- 治疗 专家 寻医问药 特邀 疾病 访谈 医院 详情 患者 主任医师

\*Topic 6

- 功效 作用 治疗 蜂蜜 食用 具有 不宜 养生 咳嗽 补血

\*Topic 7

- 眼睛 运动 按摩 肌肉 可以 疲劳 皮肤 头发 缓解 10

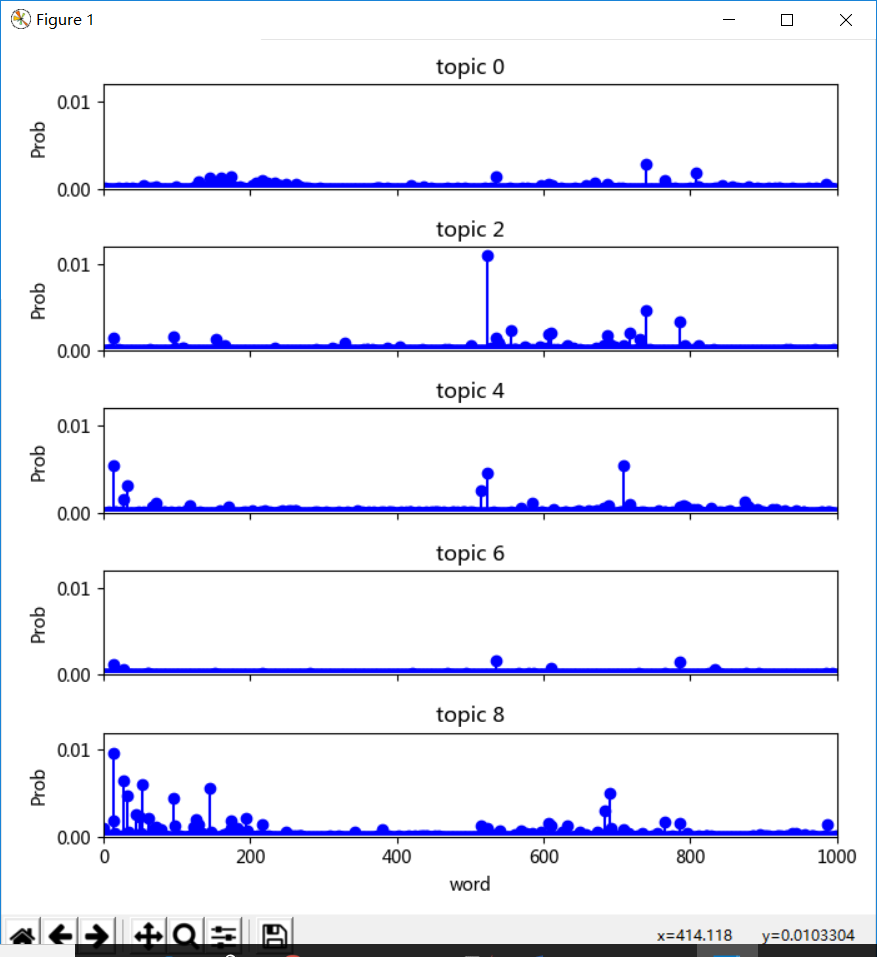
\*Topic 8

- 不要 每天 分钟 10 小时 健康 最好 时间 早餐 11

\*Topic 9

- 帮助 社区 北京 查看 删除 招聘 广告 详情 此微博 作者

（4）主题-词分布



（5）文档-主题分布

type(doc\_topic): <class 'numpy.ndarray'>

shape: (5504, 10)

[0.0047619 0.0047619 0.0047619 0.0047619 0.95714286 0.0047619

0.0047619 0.0047619 0.0047619 0.0047619 ]

doc: 0 topic: 4

[0.00769231 0.00769231 0.00769231 0.16153846 0.77692308 0.00769231

0.00769231 0.00769231 0.00769231 0.00769231]

doc: 1 topic: 4

[0.15434783 0.00217391 0.08913043 0.00217391 0.26304348 0.15434783

0.04565217 0.02391304 0.26304348 0.00217391]

doc: 2 topic: 4

[0.04565217 0.00217391 0.00217391 0.26304348 0.1326087 0.0673913

0.41521739 0.04565217 0.02391304 0.00217391]

doc: 3 topic: 6

[0.00322581 0.00322581 0.71290323 0.00322581 0.22903226 0.00322581

0.00322581 0.00322581 0.03548387 0.00322581]

doc: 4 topic: 2

[0.50227273 0.00227273 0.13863636 0.00227273 0.32045455 0.00227273

0.00227273 0.025 0.00227273 0.00227273]

doc: 5 topic: 0

[0.30465116 0.00232558 0.3744186 0.04883721 0.00232558 0.0255814

0.00232558 0.07209302 0.16511628 0.00232558]

doc: 6 topic: 2

[0.0025641 0.0025641 0.82307692 0.0025641 0.07948718 0.0025641

0.0025641 0.05384615 0.0025641 0.02820513]

doc: 7 topic: 2

[0.18684211 0.00263158 0.16052632 0.16052632 0.00263158 0.02894737

0.00263158 0.00263158 0.45 0.00263158]

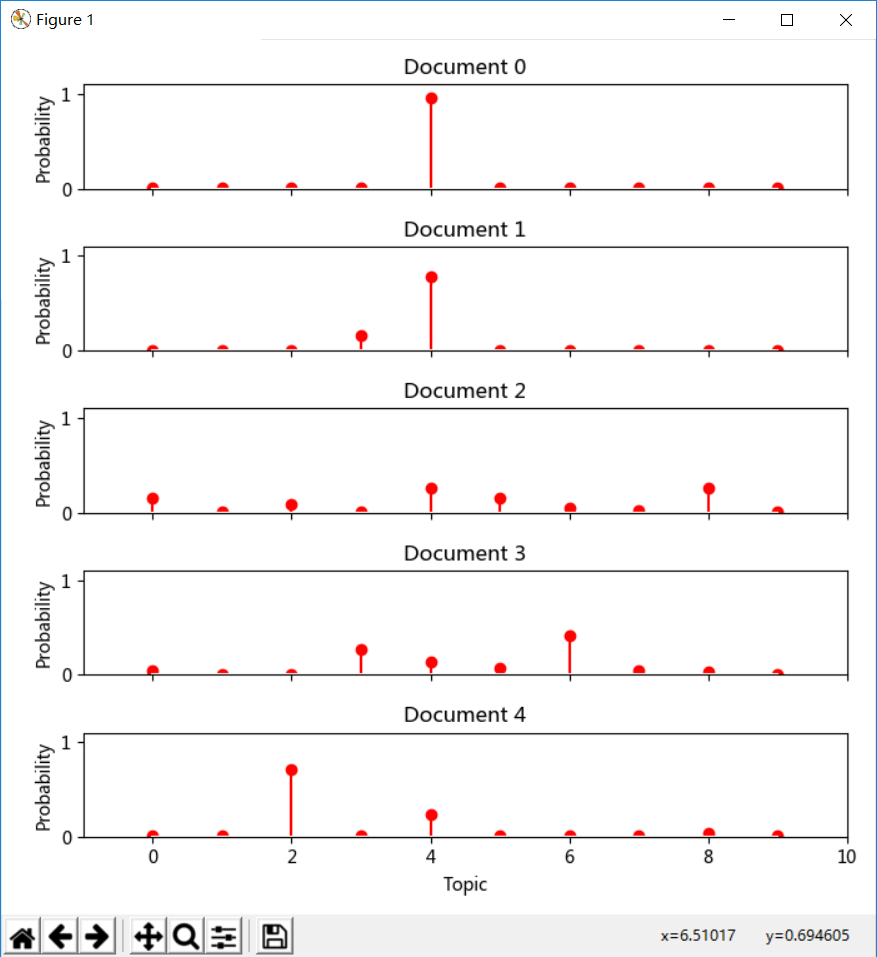
doc: 8 topic: 8

[0.00344828 0.00344828 0.27931034 0.00344828 0.62413793 0.00344828

0.03793103 0.00344828 0.03793103 0.00344828]

doc: 9 topic: 4

前10篇文档对应的主题列表： [4, 4, 4, 6, 2, 0, 2, 2, 8, 4]



四、kMeans聚类

```

from sklearn import feature\_extraction

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

if \_\_name\_\_ == "\_\_main\_\_":

corpus = []

for line in open('cut\_word\_result.txt', 'r').readlines():

corpus.append(line.strip())

#print (corpus)

vectorizer = CountVectorizer()

transformer = TfidfTransformer()

#第一个fit\_transform是计算tf-idf 第二个fit\_transform是将文本转为词频矩阵

tfidf = transformer.fit\_transform(vectorizer.fit\_transform(corpus))

word = vectorizer.get\_feature\_names() # 所有的特征词，即关键词

#print (word)

weight = tfidf.toarray()

print(len(weight))

print(len(weight[0]))

print(weight[0:10, 0:30])

print ('Features length: ' + str(len(word)))

####################聚类####################

SSE = []

for k in range(2, 21, 2):

clf = KMeans(n\_clusters = k)

s = clf.fit(weight)

print ("s:\n", s)

#中心点

print("center len:\n", len(clf.cluster\_centers\_))

print("center len of each:\n", len(clf.cluster\_centers\_[0]))

print("centers:\n", clf.cluster\_centers\_[0:20])

#每个样本所属的簇

print("labels len:\n", len(clf.labels\_))

print(clf.labels\_[0:20])

i = 0

while i <= len(clf.labels\_):

print (i, clf.labels\_[i])

i += 1

if(50 == i):

break

#用来评估簇的个数是否合适，距离越小说明簇分的越好，选取临界点的簇个数

print("inertia:\n", k, clf.inertia\_)

SSE.append(clf.inertia\_)

x = range(2, 21, 2)

plt.xlabel('k')

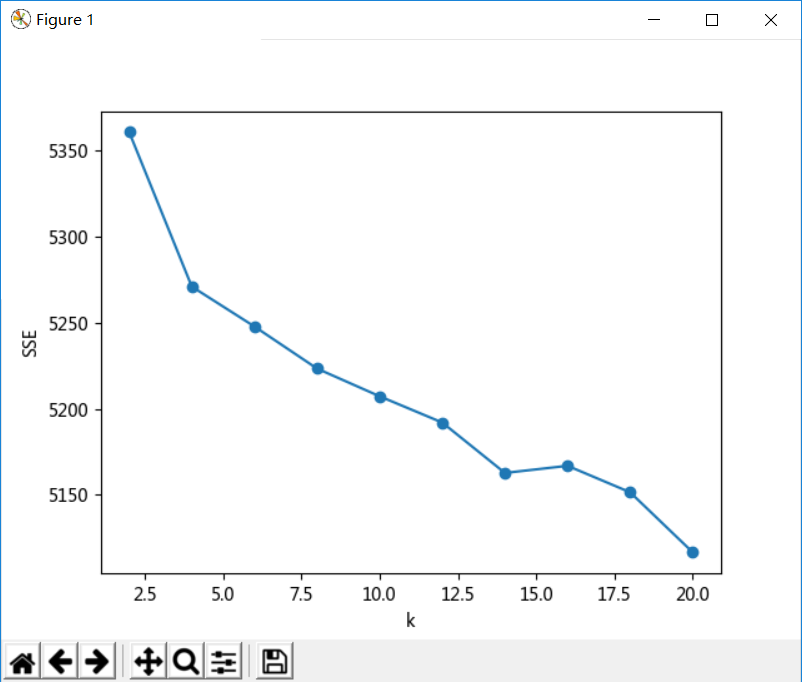
plt.ylabel('SSE')

plt.plot(x, SSE, 'o-')

plt.show()

```

运行结果：



从这个图可以看出，用肘点k = 14比较好。

五、提取关键字

六、后续问题

1 文本中有些数字，是用来表示序号的，比如“1、”, “2.”, “3 “。这些在分词的时候可以考虑去掉。

2 除了SSE，还要测试准确率、召回率和F值。

3