[python] 基于k-means和tfidf的文本聚类代码简单实现 - Eastmount的专栏 - 博客频道

分类：
Python爬虫（24）
知识图谱（14）
数据挖掘（9）

目录(?)[+]

俗话说“外行看热闹，内行看门道“，作为一个机器学习的门外汉，刚研究python机器学习scikit-learn两周时间，虽然下面这段程序可能对于那些专研算法或机器学习的人来说非常简单，但对于一些入门的同学和我自己还是非常有帮助的。如果文章中有错误或不足之处，还请你微微一笑，原谅之；当然也非常欢迎你提出建议或指正~

基本步骤包括：

1. 使用python+selenium分析dom结构爬取百度|互动百科文本摘要信息；
2. 使用jieba结巴分词对文本进行中文分词，同时插入字典关于关键词；
3. scikit-learn对文本内容进行tfidf计算并构造N*M矩阵(N个文档 M个特征词)；
4. 再使用K-means进行文本聚类(省略特征词过来降维过程)；
5. 最后对聚类的结果进行简单的文本处理，按类簇归类，也可以计算P/R/F特征值；
6. 总结这篇论文及K-means的缺点及知识图谱的一些内容。

当然这只是一篇最最基础的文章，更高深的分类、聚类、LDA、SVM、随机森林等内容，自己以后慢慢学习吧！这篇作为在线笔记，路漫漫其修远兮，fighting~

一. 爬虫实现

爬虫主要通过Python+Selenium+Phantomjs实现，爬取百度百科和互动百科旅游景点信息，其中爬取百度百科代码如下。
参考前文：[Python爬虫] Selenium获取百度百科旅游景点的InfoBox消息盒

实现原理：
首先从Tourist_spots_5A_BD.txt中读取景点信息，然后通过调用无界面浏览器PhantomJS（Firefox可替代）访问百度百科链接"http://baike.baidu.com/"，通过Selenium获取输入对话框ID，输入关键词如"故宫"，再访问该百科页面。最后通过分析DOM树结构获取摘要的ID并获取其值。核心代码如下：

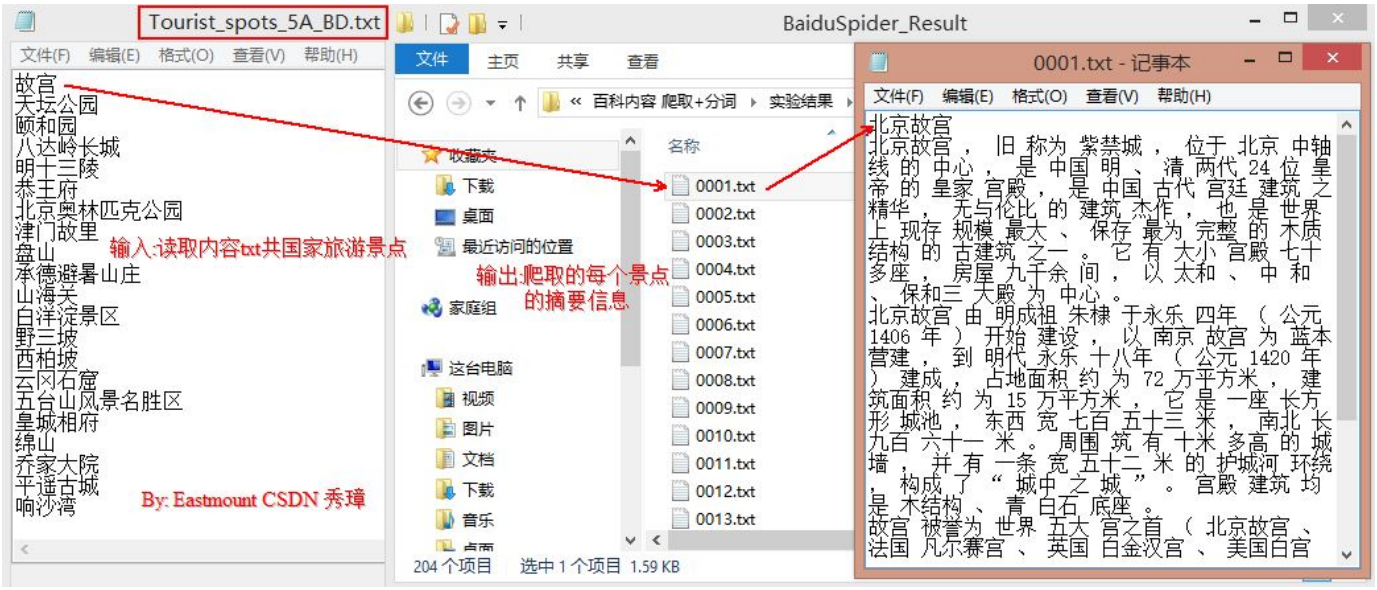
```
driver.find_elements_by_xpath("//div[@class='lemma-summary']/div")
```

PS：Selenium更多应用于自动化测试，推荐Python爬虫使用scrapy等开源工具。

```
1. #  coding=utf-8
2. """
3. Created  on  2015-09-04  @author:  Eastmount
4. """
5. import  time
6. import  re
7. import  os
8. import  sys
9. import  codecs
10. import  shutil
11. from  selenium  import  webdriver
12. from  selenium.webdriver.common.keys  import  Keys
13. import  selenium.webdriver.support.ui  as  ui
14. from  selenium.webdriver.common.action_chains  import  ActionChains
15. #Open  PhantomJS
16. driver  =  webdriver.PhantomJS(executable_path="G:\phantomjs-1.9.1-
    windows\phantomjs.exe")
17. #driver  =  webdriver.Firefox()
18. wait  =  ui.WebDriverWait(driver,10)
19. #Get  the  Content  of  5A  tourist  spots
20. def  getInfobox(entityName,  fileName):
21.         try:
22.                 #create  paths  and  txt  files
23.                 print  u'文件名称:  ',  fileName
24.                 info  =  codecs.open(fileName,  'w',  'utf-8')
25.                 #locate  input    notice:  1.visit  url  by  unicode  2.write  files
26.                 #Error:  Message:  Element  not  found  in  the  cache  -
27.                 #             Perhaps  the  page  has  changed  since  it  was  looked
      up
28.                 #解决方法:  使用Selenium和Phantomjs
29.                 print  u'实体名称:  ',  entityName.rstrip('\n')
30.                 driver.get("http://baike.baidu.com/")
31.                 elem_inp  =  driver.find_element_by_xpath("//form[@id='searchForm']/inpu
    t")
32.                 elem_inp.send_keys(entityName)
33.                 elem_inp.send_keys(Keys.RETURN)
34.                 info.write(entityName.rstrip('\n')+'\r\n')    #codecs不支持'\n'换行
35.                 time.sleep(2)
36.                 #load  content  摘要
37.                 elem_value  =  driver.find_elements_by_xpath("//div[@class='lemma-
```

```
    summary']/div")
                for value in elem_value:
                    print value.text
                    info.writelines(value.text + '\r\n')
                time.sleep(2)
        except Exception,e:        #'utf8' codec can't decode byte
                print "Error: ",e
        finally:
                print '\n'
                info.close()
#Main function
def main():
        #By function get information
        path = "BaiduSpider\\"
        if os.path.isdir(path):
                shutil.rmtree(path, True)
        os.makedirs(path)
        source = open("Tourist_spots_5A_BD.txt", 'r')
        num = 1
        for entityName in source:
                entityName = unicode(entityName, "utf-8")
                if u'故宫' in entityName:      #else add a '?'
                        entityName = u'北京故宫'
                name = "%04d" % num
                fileName = path + str(name) + ".txt"
                getInfobox(entityName, fileName)
                num = num + 1
        print 'End Read Files!'
        source.close()
        driver.close()
if __name__ == '__main__':
        main()
```

运行结果如下图所示：



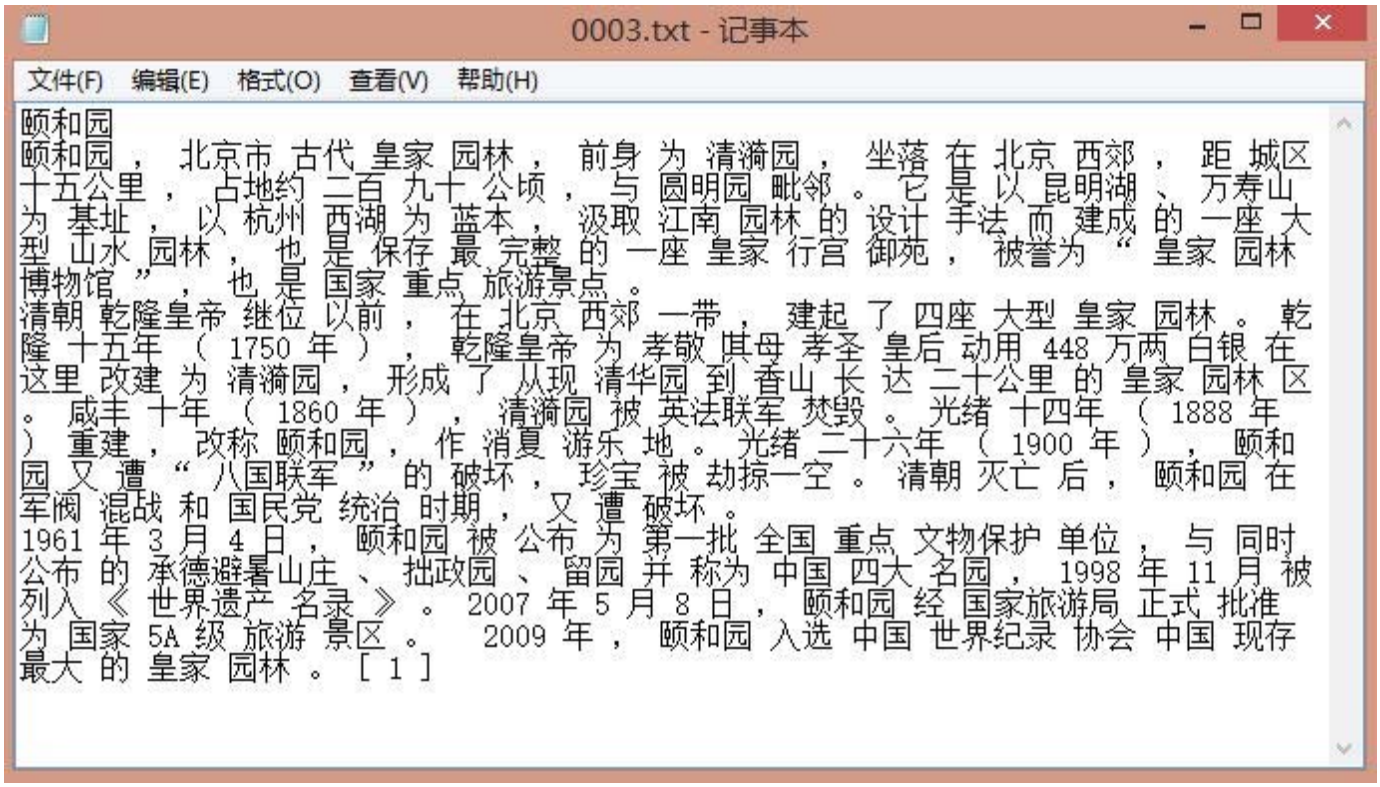
二. 中文分词中文分词主要使用的是Python+Jieba分词工具，同时导入自定义词典dict_baidu.txt，里面主要是一些专业景点名词，如"黔清宫"分词"黔/清宫"，如果词典中存在专有名词"乾清宫"就会先查找词典。

参考前文：[python] 使用Jieba工具中文分词及文本聚类概念

```
1.          #create  path
2.          path  =  "BaiduSpider\\"
3.          respath  =  "BaiduSpider_Result\\"
4.          if  os.path.isdir(respath):
5.                  shutil.rmtree(respath,  True)
6.          os.makedirs(respath)
7.          num  =  1
8.          while  num<=204:
9.                  name  =  "%04d"  %  num
10.                 fileName  =  path  +  str(name)  +  ".txt"
11.                 resName  =  respath  +  str(name)  +  ".txt"
12.                 source  =  open(fileName,  'r')
13.                 if  os.path.exists(resName):
14.                         os.remove(resName)
15.                 result  =  codecs.open(resName,  'w',  'utf-8')
16.                 line  =  source.readline()
17.                 line  =  line.rstrip('\n')
18.                 while  line!="":
19.                         line  =  unicode(line,  "utf-8")
20.                         seglist  =  jieba.cut(line,cut_all=False)     #精确模式
21.                         output  =  '  '.join(list(seglist))                  #空格拼
    接
22.                         print  output
23.                         result.write(output  +  '\r\n')
24.                         line  =  source.readline()
25.                 else:
```

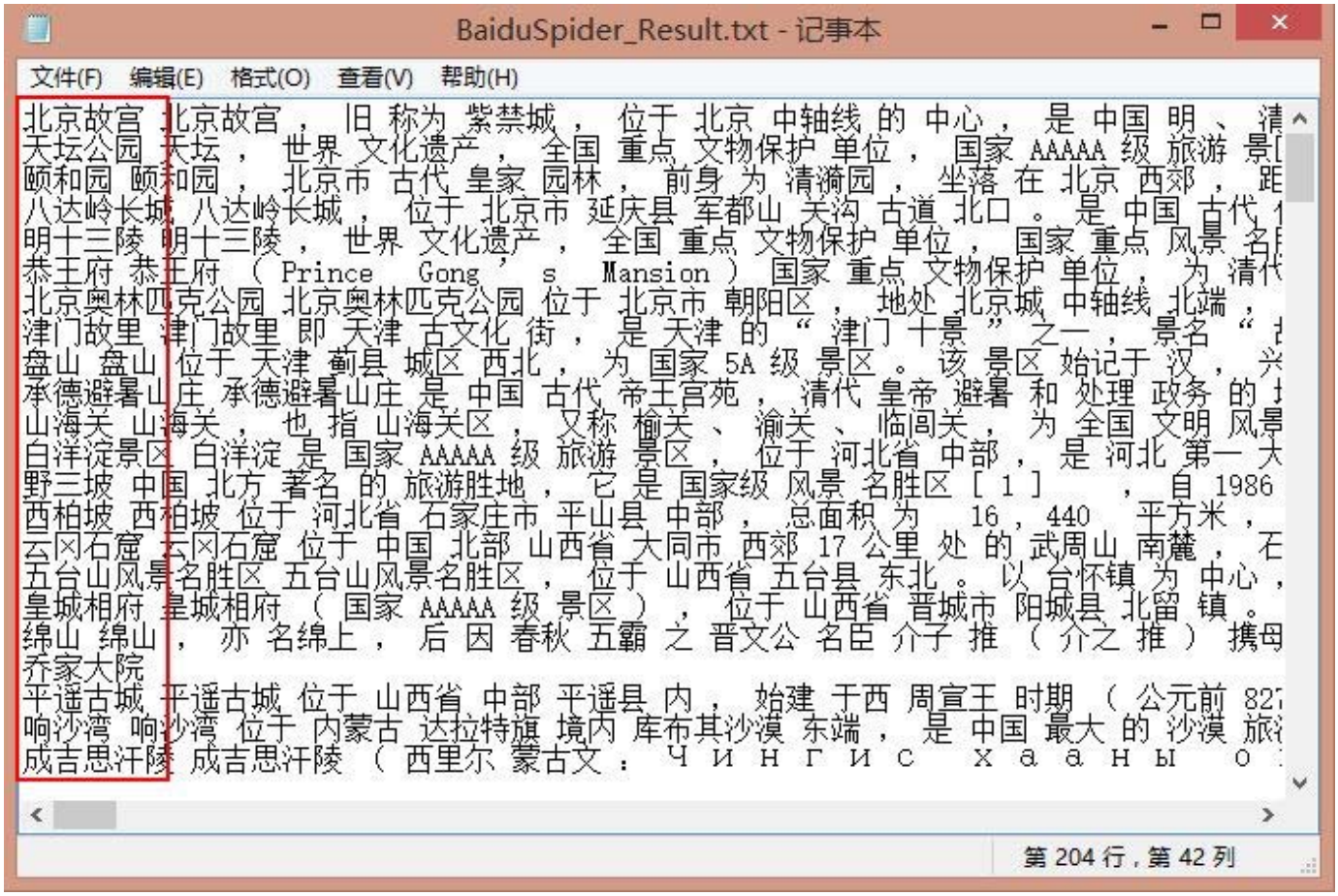
```
                    print 'End file: ' + str(num)
                    source.close()
                    result.close()
                num = num + 1
        else:
                print 'End All'
#Run function
if __name__ == '__main__':
        read_file_cut()
```

按照Jieba精确模式分词且空格拼接，"0003.txt 颐和园"分词结果如下图所示：



为方便后面的计算或对接一些sklearn或w2v等工具，下面这段代码将结果存储在同一个txt中，每行表示一个景点的分词结果。

每行一个景点的分词结果，运行结果如下图所示：



三. 计算TF-IDF

此时，需要将文档相似度问题转换为数学向量矩阵问题，可以通过VSM向量空间模型来存储每个文档的词频和权重，特征抽取完后，因为每个词语对实体的贡献度不同，所以需要对这些词语赋予不同的权重。计算词项在向量中的权重方法——TF-IDF。

相关介绍：

它表示TF（词频）和IDF（倒文档频率）的乘积：

$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

其中TF表示某个关键词出现的频率，IDF为所有文档的数目除以包含该词语的文档数目的对数值。

$$IDF = \log_2 \frac{|D|}{|w \in d|}$$

|D|表示所有文档的数目，|w∈d|表示包含词语w的文档数目。

最后TF-IDF计算权重越大表示该词条对这个文本的重要性越大，它的目的是去除一些"的、了、等"出现频率较高的常用词。

参考前文：Python简单实现基于VSM的余弦相似度计算

基于VSM的命名实体识别、歧义消解和指代消解

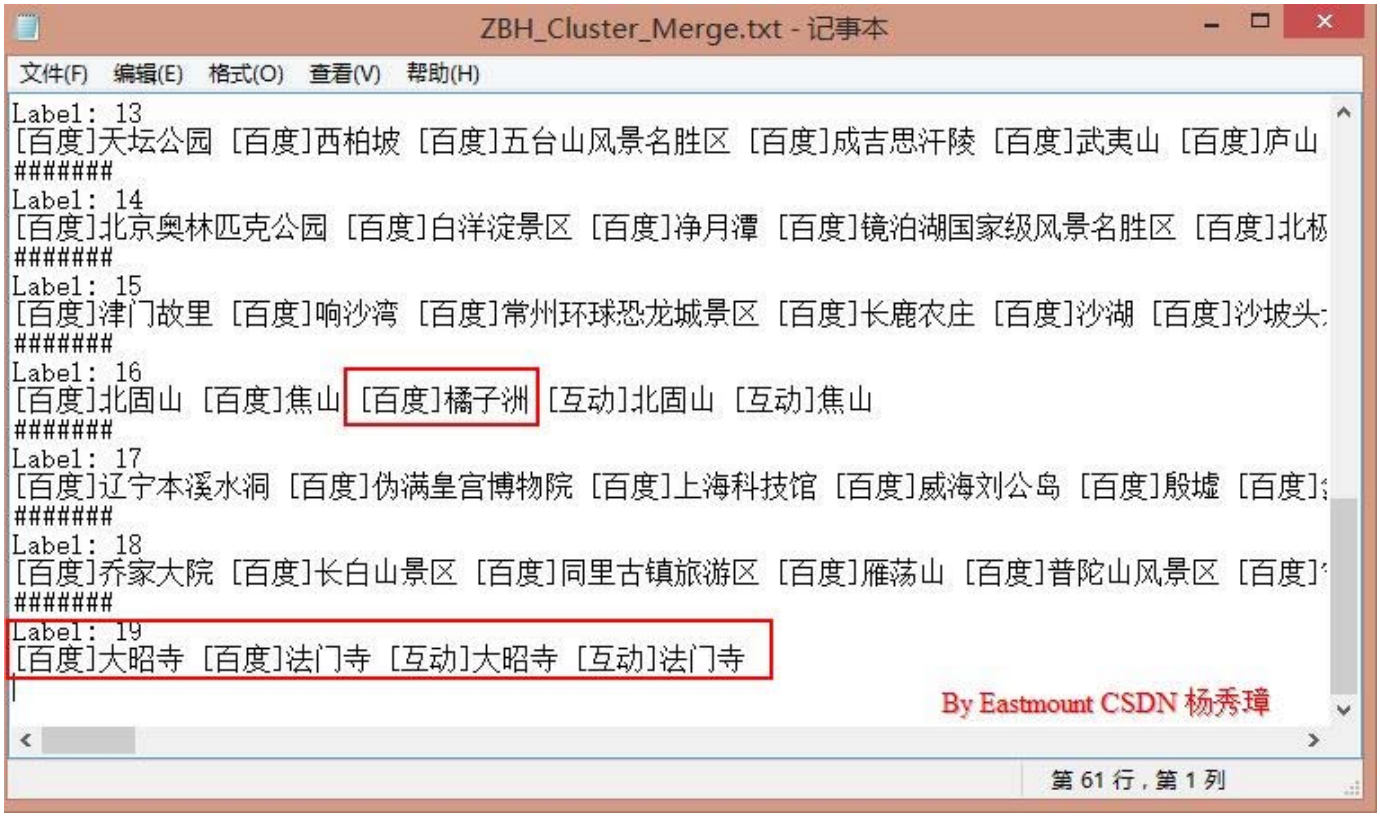
下面是使用scikit-learn工具调用CountVectorizer()和TfidfTransformer()函数计算TF-IDF值，同时后面"四.K-means聚类"代码也包含了这部分，该部分代码先提出来介绍。

```
1. #  coding=utf-8
```

```
import  os
import  sys
import  codecs
'''''
@2016-01-07  By  Eastmount
功能:合并实体名称和聚类结果  共类簇20类
输入:BH_EntityName.txt  Cluster_Result.txt
输出:ZBH_Cluster_Merge.txt  ZBH_Cluster_Result.txt
'''
source1  =  open("BH_EntityName.txt",'r')
source2  =  open("Cluster_Result.txt",'r')
result1  =  codecs.open("ZBH_Cluster_Result.txt",  'w',  'utf-8')
#############################################################################
#                                                    第一部分  合并实体名称和类簇
lable  =  []              #存储408个类标  20个类
content  =  []          #存储408个实体名称
name  =  source1.readline()
#总是多输出空格  故设置0  1使其输出一致
num  =  1
while  name!="":
        name  =  unicode(name.strip('\r\n'),  "utf-8")
        if  num  ==  1:
                res  =  source2.readline()
                res  =  res.strip('\r\n')
                value  =  res.split('  ')
                no  =  int(value[0])  -  1      #行号
                va  =  int(value[1])              #值
                lable.append(va)
                content.append(name)
                print  name,  res
                result1.write(name  +  '  '  +  res  +  '\r\n')
                num  =  0
        elif  num  ==  0:
                num  =  1
        name  =  source1.readline()
else:
        print  'OK'
        source1.close()
        source2.close()
        result1.close()
#测试输出  其中实体名称和类标一一对应
i  =  0
```

```
while  i  <  len(lable):
        print  content[i],  (i+1),  lable[i]
        i  =  i  +  1
#############################################################################
#                                                第二部分  合并类簇  类1  .....  类2  .....
#定义定长20字符串数组  对应20个类簇
output  =  ['']*20
result2  =  codecs.open("ZBH_Cluster_Merge.txt",  'w',  'utf-8')
#统计类标对应的实体名称
i  =  0
while  i  <  len(lable):
        output[lable[i]]  +=  content[i]  +  '  '
        i  =  i  +  1
#输出
i  =  0
while  i  <  20:
        print  '#######'
        result2.write('#######\r\n')
        print  'Label:  '  +  str(i)
        result2.write('Label:  '  +  str(i)  +  '\r\n')
        print  output[i]
        result2.write(output[i]  +  '\r\n')
        i  =  i  +  1
result2.close()
```

输出结果如下图所示，其中label19可以发现百度百科和互动百科的"大昭寺、法门寺"文本内容都划分为一类，同时也会存在一些错误的类别，如Label15中的"橘子洲"。



PS：如果你想进行准确率、回归率、F特征值比较，可以进一步去学习sklearn官方文档。通常的文本数据集的类标如"教育、体育、娱乐"，把不同内容的新闻聚在一类，而这个略有区别，它主要是应用于我实际的毕设。

六. 总结与不足这篇文章更多的是一些基础内容的代码实现，可能对一些初学者有用，同时也是我的在线笔记吧！主要内容包括：

1. python+selenium爬取
2. jieba中文分词
3. sklearn+tfidf矩阵权重计算
4. kmeans简单实现及结果对比

Kmeans聚类是一种自下而上的聚类方法，它的优点是简单、速度快；缺点是聚类结果与初始中心的选择有关系，且必须提供聚类的数目。

Kmeans的第二个缺点是致命的，因为在有些时候，我们不知道样本集将要聚成多少个类别，这种时候kmeans是不适合的，推荐使用hierarchical 或meanshift来聚类。第一个缺点可以通过多次聚类取最佳结果来解决。

推荐一些关于Kmeans及实验评估的文章：

浅谈Kmeans聚类 - easymind223
基于K—Means的文本聚类（强推基础介绍） - freesum
基于向量空间模型的文本聚类算法 - helld123
KMeans文档聚类python实现（代码详解） - skineffect
Kmeans文本聚类系列之全部C++代码 - finallyliuyu
文本聚类—kmeans - zengkui111

不论如何，最后还是希望文章对你有所帮助！深夜写文不易，且看且珍惜吧~
(By:Eastmount 2016-01-08 深夜3点 http://blog.csdn.NET//eastmount/)

顶
9

回复dongfenpt21638：你好，你可以自己定义一些存在的景点信息，存储在Tourist_5A_BD.txt中，其中每行一个景点名，再进行爬取，如下所示：
故宫
天坛公园
颐和园
八达岭长城
明十三陵
恭王府
北京奥林匹克公园
津门故里
盘山
承德避暑山庄
山海关
白洋淀景区
野三坡
西柏坡
云冈石窟
五台山风景名胜区
皇城相府
绵山
乔家大院