【python数据挖掘课程】二十四.KMeans文本聚类分析互动百科语料

这是《Python数据挖掘课程》系列文章，也是我上课内容及书籍中的一个案例。本文主要讲述文本聚类相关知识，包括中文分词、数据清洗、特征提取、TF-IDF、KMeans聚类等步骤。

本篇文章为基础性文章，希望对你有所帮助，提供些思路，也是自己教学的内容。如果文章中存在错误或不足之处，还请海涵。同时，推荐大家阅读我以前的文章了解其他知识。

前文参考：

【Python数据挖掘课程】一.安装Python及爬虫入门介绍
【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制
【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐
【Python数据挖掘课程】九.回归模型LinearRegression简单分析氧化物数据
【python数据挖掘课程】十.Pandas、Matplotlib、PCA绘图实用代码补充
【python数据挖掘课程】十一.Pandas、Matplotlib结合SQL语句可视化分析
【python数据挖掘课程】十二.Pandas、Matplotlib结合SQL语句对比图分析
【python数据挖掘课程】十三.WordCloud词云配置过程及词频分析
【python数据挖掘课程】十四.Scipy调用curve_fit实现曲线拟合
【python数据挖掘课程】十五.Matplotlib调用imshow()函数绘制热图
【python数据挖掘课程】十六.逻辑回归LogisticRegression分析鸢尾花数据
【python数据挖掘课程】十七.社交网络Networkx库分析人物关系（初识篇）
【python数据挖掘课程】十八.线性回归及多项式回归分析四个案例分享
【python数据挖掘课程】十九.鸢尾花数据集可视化、线性回归、决策树花样分析
【python数据挖掘课程】二十.KNN最近邻分类算法分析详解及平衡秤TXT数据集读取

【python数据挖掘课程】二十一.朴素贝叶斯分类器详解及中文文本舆情分析
【python数据挖掘课程】二十二.Basemap地图包安装入门及基础知识讲解
【python数据挖掘课程】二十三.时间序列金融数据预测及Pandas库详解

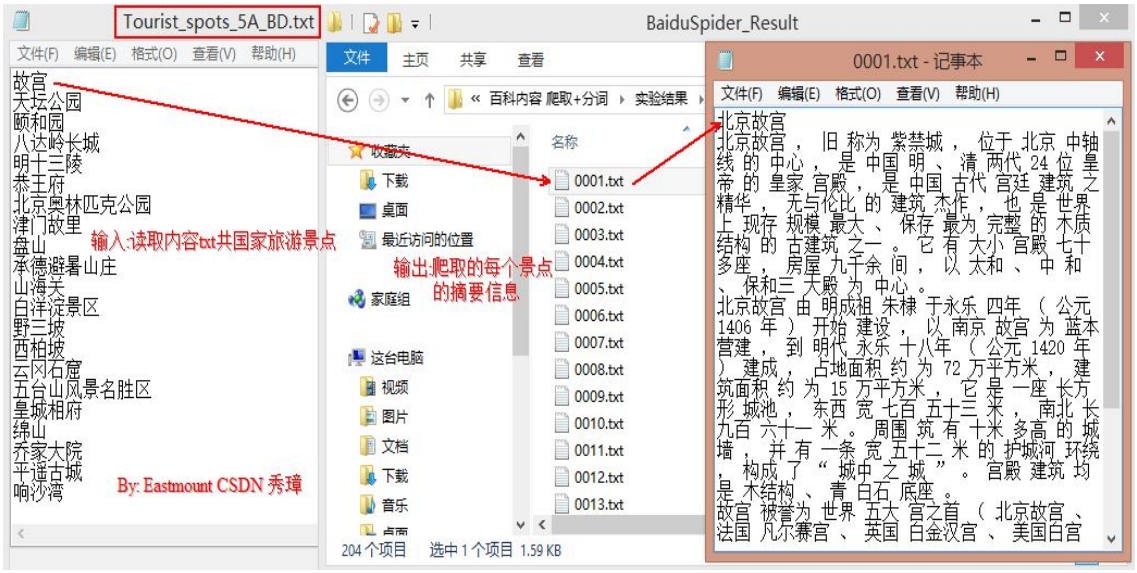
这篇文章代码和实验分析为主，不进行详细讲解，详见：
[python] 基于k-means和tfidf的文本聚类代码简单实现
[python] Kmeans文本聚类算法+PAC降维+Matplotlib显示聚类图像

PSS：最近参加CSDN2018年博客评选，希望您能投出宝贵的一票。我是59号，Eastmount，杨秀璋。投票地址：https://bss.csdn.net/m/topic/blog_star2018/index

五年来写了314篇博客，12个专栏，是真的热爱分享，热爱CSDN这个平台，也想帮助更多的人，专栏包括Python、数据挖掘、网络爬虫、图像处理、C#、Android等。现在也当了两年老师，更是觉得有义务教好每一个学生，让贵州学子好好写点代码，学点技术，"师者，传到授业解惑也"，提前祝大家新年快乐。2019我们携手共进，为爱而生。

一. Python文本抓取

爬虫主要通过Python+Selenium+Phantomjs实现，爬取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
爬取的数据集如下图所示，互动百科的每类主题各100篇网页文本，涉及人物明星、旅游景区、动物、世界国家四个主题。运行结果如下图所示：



实现原理：

我们首先分析互动百科搜索词条的一些规则，比如搜索人物“贵州”，对应的超链为“http://www.baike.com/wiki/贵州”，对应页面如图9.16所示，从图中可以看到，顶部的超链接URL、词条为“贵州”、第一段为“贵州”的摘要信息、“右边为对应的图片等信息。

同理，搜索编程语言“Python”，对应的超链接为“http://www.baike.com/wiki/Python”，可以得出一个简单的规则，即“http://www.baike.com/wiki/词条”可以搜索对应的知识，如编程语言“Java”对应的“http://www.baike.com/wiki/Java&prd=button_doc_entry”，这里定义了搜索方式，它是通过点击按钮“进入词条”进行搜索的，省略“prd=button_doc_entry”参数同样可以得到相同的结果。

然后，需要分布获取这十门语言的摘要信息。在浏览器中选中摘要部分，右键鼠标点击“审查元素”返回结果如图9.18所示，可以在底部看到摘要部分对应的HTML源代码。

"Python" 词条摘要部分对应的HTML核心代码如下所示：

```
<div class="summary" name="anchor" id="anchor">
<p>Python（英语发音：/'paɪθən/），是一种面向对象、解释型计算机程序设计语言，由Guido van
Rossum于1989年发明...</p>
  <span><a action="editsummaryhref" href="javascript:void(0);"
onclick="editSummary();return false;"> 编辑摘要
</a></span>
</div>
```

调用Selenium的find_element_by_xpath("//summary[@class='summary']/p")函数，可以获取摘要段落信息，核心代码如下。

```
driver = webdriver.Firefox()
                                url = "http://www.baike.com/wiki/" + name
driver.get(url)
elem = driver.find_element_by_xpath("//div[@class='summary']/p")
print elem.text
```

这段代码的基本步骤是:

1.首先调用webdriver.Firefox()驱动，打开火狐浏览器。

2.分析网页超链接，并调用driver.get(url)函数访问。

3.分析网页DOM树结构，调用driver.find_element_by_xpath()进行分析。

4.输出结果，部分网站的内容需要存储至本地，并且需要过滤掉不需要的内容等。

下面是完整的代码及详细讲解，参考自己的书籍《Python网络数据爬取及分析从入门到精通（爬取篇）》。

```
# coding=utf-8
# test09_03.py
import os
import codecs
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox()

#获取摘要信息
def getAbstract(name):
    try:
        #新建文件夹及文件
        basePathDirectory = "Hudong_Coding"
        if not os.path.exists(basePathDirectory):
            os.makedirs(basePathDirectory)
        baiduFile = os.path.join(basePathDirectory,"HudongSpider.txt")
        #文件不存在新建,存在则追加写入
        if not os.path.exists(baiduFile):
            info = codecs.open(baiduFile,'w','utf-8')
        else:
            info = codecs.open(baiduFile,'a','utf-8')

        url = "http://www.baike.com/wiki/" + name
        print url
        driver.get(url)
        elem = driver.find_element_by_xpath("//div[@class='summary']/p")
        print elem.text
```

```
            info.writelines(elem.text+'\r\n')
        except Exception,e:
            print "Error: ",e
        finally:
            print '\n'
            info.write('\r\n')

    #主函数
    def main():
        languages = ["JavaScript", "Java", "Python", "Ruby", "PHP",
                     "C++", "CSS", "C#", "C", "GO"]
        print u'开始爬取'
        for lg in languages:
            print lg
            getAbstract(lg)
        print u'结束爬取'

    if __name__ == '__main__':
        main()
```

输出如下图所示，将数据集 换成对应的景区即可。

```
JavaScript
http://www.baike.com/wiki/JavaScript
JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解
释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通
用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。在1995年时，由Netscape公司
的Brendan Eich，在网景导航者浏览器上首次设计实现而成。因为Netscape与Sun合作，Netscape管
理层希望它外观看起来像Java，因此取名为JavaScript。但实际上它的语法风格与Self及Scheme较为
接近。为了取得技术优势，微软推出了JScript，CEnvi推出ScriptEase，与JavaScript同样可在浏览
器上运行。为了统一规格，因为JavaScript兼容于ECMA标准，因此也称为ECMAScript。


Java
http://www.baike.com/wiki/Java
Java是一种可以撰写跨平台应用软件的面向对象的程序设计语言，是由Sun Microsystems公司于1995
年5月推出的Java程序设计语言和Java平台（即JavaEE，JavaME，JavaSE）的总称。Java自面世后就
非常流行，发展迅速，对C++语言形成了有力冲击。Java技术具有卓越的通用性、高效性、平台移植性和
安全性，广泛应用于个人PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全
球最大的开发者专业社群。在全球云计算和移动互联网的产业环境下，Java更具备了显著优势和广阔前景
。Java是目前世界上流行的计算机编程语言，是一种可以撰写跨平台应用软件的面向对象的程序设计语言
。全球有25亿Java器运行着Java，450多万Java开发者活跃在地球的每个角落，数以千万计的Web用户
每次上网都亲历Java的威力。
```

二. 数据预处理

由于爬取的数据集都是分布于四个主题文件夹中，每个文件中共100个，所以需要将这四个主题的文本合并成一个txt文件，其代码如下：

```
# coding=utf-8
import re
import os
import sys
import codecs
import shutil

def merge_file():
    path = "BaiduSpiderSpots\\"
    resName = "BaiduSpider_Result.txt"
    if os.path.exists(resName):
        os.remove(resName)
    result = codecs.open(resName, 'w', 'utf-8')

    num = 1
    while num <= 100:
        name = "%04d" % num
        fileName = path + str(name) + ".txt"
        source = open(fileName, 'r')
        line = source.readline()
        line = line.strip('\n')
        line = line.strip('\r')

        while line!="":
            line = unicode(line, "utf-8")
            line = line.replace('\n',' ')
            line = line.replace('\r',' ')
            result.write(line+ ' ')
            line = source.readline()
        else:
            print 'End file: ' + str(num)
            result.write('\r\n')
            source.close()
        num = num + 1

    else:
        print 'End All'
        result.close()
```

```
if __name__ == '__main__':
                                merge_file()
```

合并后的结果如下图所示，生成的HudongSpider_Result.txt共400个，1-100为动物、101-200为景区、201-300位人物、301-400为国家，每一行表示所爬取的一个网页文本。

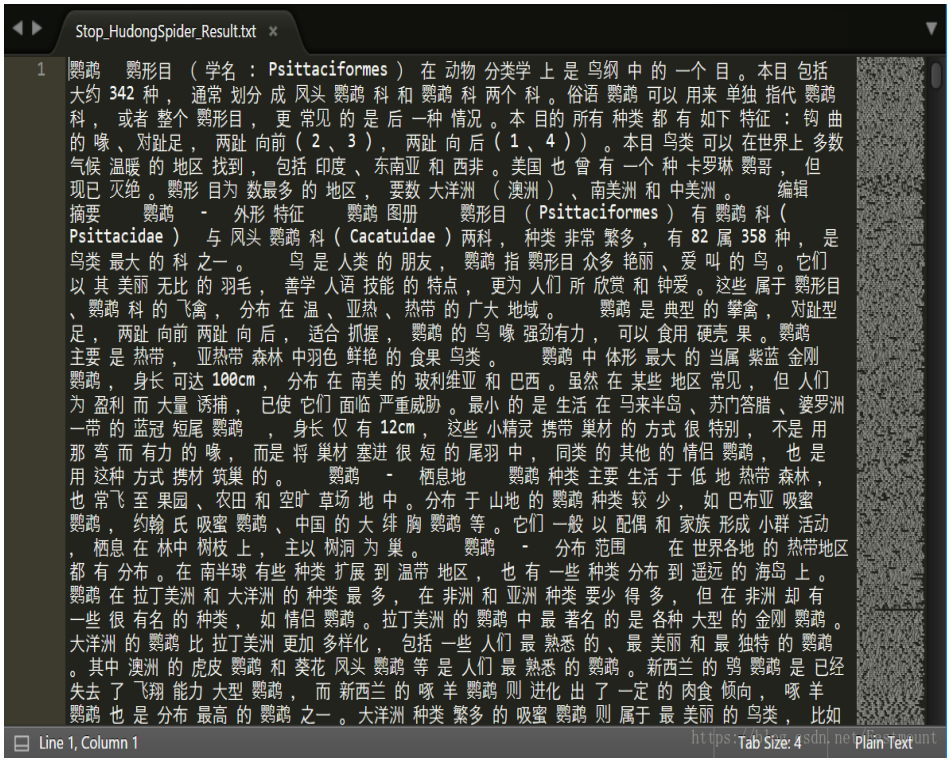
三. 中文分词

中文分词主要使用的是Python+Jieba分词工具，代码如下：

```
#encoding=utf-8
import sys
import re
import codecs
import os
import shutil
import jieba
import jieba.analyse
```

```
  #导入自定义词典
#jieba.load_userdict("dict_baidu.txt")

#Read file and cut
def read_file_cut():

    fileName = "HudongSpider_Result.txt"
    source = open(fileName, 'r')
    resName = "Stop_HudongSpider_Result.txt"
    result = codecs.open(resName, 'w', 'utf-8')
    line = source.readline()

    while line!="":
        line = unicode(line, "utf-8")
        seglist = jieba.cut(line,cut_all=False)  #精确模式
        output = ' '.join(list(seglist))         #空格拼接
        #print output
        result.write(output)
        line = source.readline()
    else:
        source.close()
        result.close()
        print 'End All'

#Run function
if __name__ == '__main__':
    read_file_cut()sc
```

输出结果如下图所示，采用空格连接，同时可以导入词典进行分词。分词之后，也可以进行停用词过滤、特殊符号去除等数据清洗，这里不再介绍，详见前文。



四. KMeans聚类分析

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，同时调用sklearn.cluster中的KMeans算法进行文本聚类。

完整代码：

```
# coding=utf-8
import time
import re
import os
import sys
import codecs
import shutil
import numpy as np
import matplotlib
import scipy
import matplotlib.pyplot as plt
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import HashingVectorizer

if __name__ == "__main__":

    ##########################################################################
```

```
#                                  第一步 计算TFIDF
#文档预料 空格连接
corpus = []

#读取预料 一行预料为一个文档
for line in open('Stop_HudongSpider_Result.txt', 'r').readlines():
    #print line
    corpus.append(line.strip())
#print corpus

#参考: http://blog.csdn.net/abcjennifer/article/details/23615947
#vectorizer = HashingVectorizer(n_features = 4000)

#将文本中的词语转换为词频矩阵 矩阵元素a[i][j] 表示j词在i类文本下的词频
vectorizer = CountVectorizer()

#该类会统计每个词语的tf-idf权值
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))

#获取词袋模型中的所有词语
word = vectorizer.get_feature_names()

#将tf-idf矩阵抽取出来，元素w[i][j]表示j词在i类文本中的tf-idf权重
weight = tfidf.toarray()

#打印特征向量文本内容
print 'Features length: ' + str(len(word))
resName = "BHTfidf_Result.txt"
result = codecs.open(resName, 'w', 'utf-8')
for j in range(len(word)):
    result.write(word[j] + ' ')
result.write('\r\n\r\n')

#打印每类文本的tf-idf词语权重，第一个for遍历所有文本，第二个for便利某一类文本下的词语权
重
for i in range(len(weight)):
    #print u"-------这里输出第", i, u"类文本的词语tf-idf权重------"
    for j in range(len(word)):              #print weight[i][j],
        result.write(str(weight[i][j]) + ' ')
    result.write('\r\n\r\n')

result.close()
```

```
  ##############################################################################
#                                   第二步 聚类Kmeans

print 'Start Kmeans:'
from sklearn.cluster import KMeans
clf = KMeans(n_clusters=4)   #景区 动物 人物 国家
s = clf.fit(weight)
print s

'''
print 'Start MiniBatchKmeans:'
from sklearn.cluster import MiniBatchKMeans
clf = MiniBatchKMeans(n_clusters=20)
s = clf.fit(weight)
print s
'''

#中心点
print(clf.cluster_centers_)

#每个样本所属的簇
label = []                #存储400个类标
print(clf.labels_)
i = 1
while i <= len(clf.labels_):
    print clf.labels_[i-1]
    label.append(clf.labels_[i-1])
    i = i + 1

#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  958.137281791
print(clf.inertia_)
y_pred = clf.labels_



##############################################################################
#                                   第三步 图形输出 降维

from sklearn.decomposition import PCA
pca = PCA(n_components=2)             #输出两维
newData = pca.fit_transform(weight)   #载入N维
print newData

x = [n[0] for n in newData]
y = [n[1] for n in newData]
```

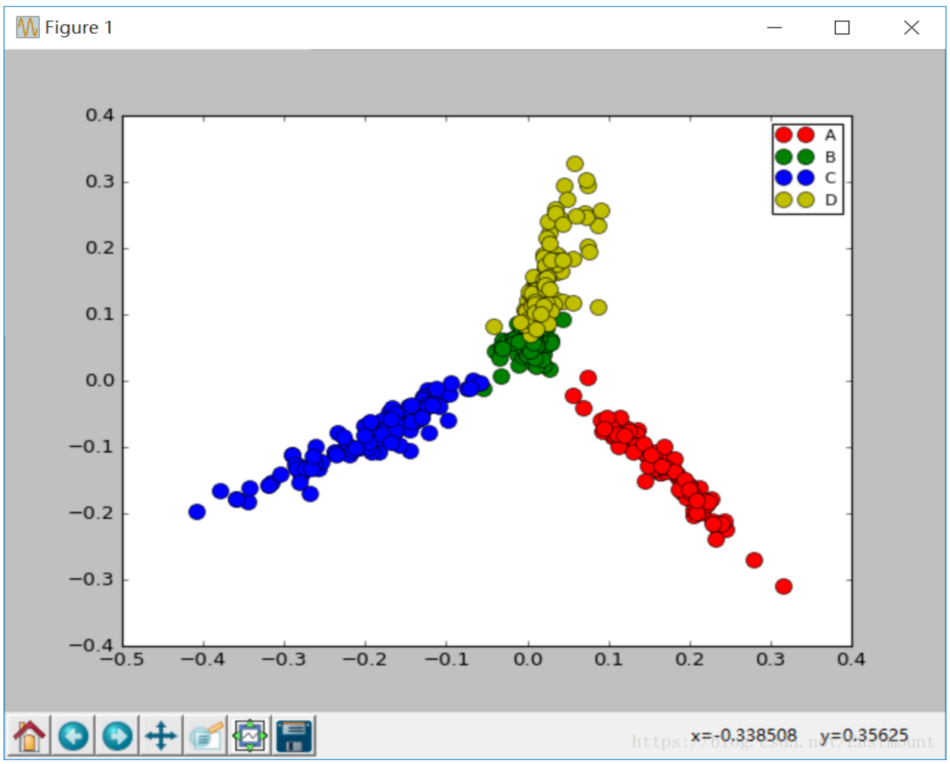
```
  x1, y1 = [], []
x2, y2 = [], []
x3, y3 = [], []
x4, y4 = [], []

#分布获取类标为0、1、2、3的数据 赋值给(x1,y1) (x2,y2) (x3,y3) (x4,y4)
i = 0
while i < len(newData):
    if y_pred[i]==0:
        x1.append(newData[i][0])
        y1.append(newData[i][1])
    elif y_pred[i]==1:
        x2.append(newData[i][0])
        y2.append(newData[i][1])
    elif y_pred[i]==2:
        x3.append(newData[i][0])
        y3.append(newData[i][1])
    elif y_pred[i]==3:
        x4.append(newData[i][0])
        y4.append(newData[i][1])
    i = i + 1

#四种颜色 红 绿 蓝，marker='x'表示类型，o表示圆点 *表示星型 x表示点
plot1, = plt.plot(x1, y1, 'or', marker="o", markersize=10)
plot2, = plt.plot(x2, y2, 'og', marker="o", markersize=10)
plot3, = plt.plot(x3, y3, 'ob', marker="o", markersize=10)
plot4, = plt.plot(x4, y4, 'oy', marker="o", markersize=10)
#plt.title("K-Means Text Clustering")  #绘制标题
plt.legend((plot1, plot2, plot3, plot4), ('A', 'B', 'C', 'D'), fontsize=10)

#四种颜色 红 绿 蓝 黑
#plt.scatter(x1, x2, c=clf.labels_,  s=100)
plt.show()
```

输出结果如下所示，将400个网页文本聚集成4类（类标0~3），每一类表示人物、景区、国家、动物。

```
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
```

```
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 0 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 0 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

同时绘制相关图形如下所示，注意散点图类标的绘制。



五. 结果评价

这里我想结合文本聚类简单叙述下最常用的评估方法：

正确率 Precision = 正确识别的个体总数 / 识别出的个体总数

召回率 Recall = 正确识别的个体总数 / 测试集中存在的个体总数

F值 F-measure = 正确率 * 召回率 * 2 / (正确率 + 召回率)

由于"clf.labels_"会返回聚类每个样本所属的簇，比如400行数据，就会返回400个label值。同时，clf = KMeans(n_clusters=4)设置了类簇为4，故每个值对应在0、1、2、3中的一个，统计结果如下：

```
动物：99识别正确、共识别100个
国家：97识别正确、共识别97个
人物：98识别正确、共识别98个
景区：99识别正确、共识别105个
```

其中以景区为例，识别出的label3数目为105，同时正确识别出的个体数=99，总共测试样本共100个，故：

准确率=99/105=0.9429

召回率=99/100=0.9900

F值=(2*0.9429*0.9900)/(0.9429+0.9900)=0.9659

最终输出结果如下所示：

主题	评价指标	传统 K-Means 文本聚类方法
旅游景区	准确率	0.9429
	召回率	0.9900
	F-值	0.9659
保护动物	准确率	0.9900
	召回率	0.9900
	F-值	0.9900
人物明星	准确率	1.000
	召回率	0.9800
	F-值	0.9899
国家地理	准确率	1.000
	召回率	0.9700
	F-值	0.9848

同时可以计算宏平均聚类准确率（Macro-Prec）和宏平均召回率（Macro-Rec）。

希望基础性文章对您有所帮助，如果文章中有错误或不足之处还请海涵。
最后推荐作者的最新出版书籍：

本书主要包括上下两册：

《Python网络数据爬取及分析从入门到精通（爬取篇）》
《Python网络数据爬取及分析从入门到精通（分析篇）》

(By:Eastmount 2018-07-06 深夜8点 http://blog.csdn.net/eastmount/)

点赞 3 收藏 分享 ...

Eastmount 博客专家

发布了444 篇原创文章 · 获赞 5907 · 访问量 484万+

他的留言板 关注