三国演义文本分析

# 项目目标

1. 对三国演义进行词频分析（主要是人名），找出出现最多的几个，并构建词云。
2. 浅层面分析各个人物之间的关系。

# 项目结构

1、wordcloud.py用于生成词云，该文件实现了基于文本的分词和统计，并生成词云。

2、Relationship.py用于从小说中找出人物之间的关联度，并给以评判。它生成了两个文件

Sanguo\_node.txt和sanguo\_edge.txt，本来是用作gephi画关系图使用。但是因为第一次用这个软件，有一些问题没能解决，生成的图像不是很清晰。所以最后采用从生成的文本中查询数据的方式(只用到了sanguo\_edge.txt)。

3、search.py用于从得到的数据中查询想要的信息。

4、data文件夹存放本次项目的一些必须数据。

# 代码介绍

1. **分析词频并构建词云**
2. 分词

如果是粗略的分词可以直接使用jieba.cut来实现，但是这样往往效果不是很好。

所以我们添加百度的提供的中文停用词数据集，先把停用词这样的无关数据从

数据集里删去，然后再分词。

stopwords = pd.read\_csv('./data/cn\_stopwords.txt', engine='python',

encoding='utf-8', names=['stopwords'])['stopwords'].tolist()

book = open('./data/sanguo-utf8.txt', encoding='utf-8').read()

'''

分词

'''

def words\_cut(book):

words=list(jieba.cut(book))

seg = set(words) - set(stopwords)

result = [i for i in words if i in seg]

return result

1. 绘图

直接调用wordcloud函数绘制，为了方便使用，封装了如下函数，

主要是对显示的词源的数量加以控制，来更好的显示效果。然后使用图像库

PIT来展示生成的图片。

def wordcloudplot(txt, maxWords):

path='c:/Windows/Fonts/simhei.ttf'

alice\_mask = np.array(PIL.Image.open('./data/zhu.png'))

wordcloud=WordCloud(font\_path=path,

background\_color="white",

margin=5, width=1800, height=800,

mask=alice\_mask,

max\_words=maxWords,

max\_font\_size=60,

random\_state=42)

wordcloud=wordcloud.generate(txt)

wordcloud.to\_file("./picture.jpg")

plt.imshow(wordcloud)

plt.axis("off")

plt.show()

1. **从小说中生成人物的关系数据**

和上面一样，同样是先分词，但是这一次的侧重点不同。上面是对三国演义中的除取停用词之外的几乎所有的词汇一并计入统计，但是这次是只对部分人名进行统计。

1. 得到姓名列表

f1 = open('./data/name.txt',encoding='utf-8')

for line in f1.readlines():

all\_names.append(line.strip().strip('\ufeff'))

for name in all\_names:

jieba.add\_word(name)

1. 得到文本文档中每一行（实际为内容的一段）中出现的人名列表。、

f2 = open('./data/sanguo-utf8.txt',encoding='utf-8')

for line in f2.readlines():

seg\_list = jieba.cut(line)

linenames.append([])

for i in seg\_list:

if i in all\_names:

linenames[-1].append(i)

if names.get(i) is None:

names[i] = 0

relationships[i] = {}

names[i] +=1

1. 对每一行的人名积分，如果两个人名同时再一段内出现，则即一分。分数越高反应两者联系越密切。

for line in linenames:

for name1 in line:

for name2 in line:

if name1 == name2:

continue

if relationships[name1].get(name2) is None:

relationships[name1][name2]=1

else:

relationships[name1][name2] += 1

1. 将数据写入文件，并按照统计的分数来进行评估。

import codecs

with codecs.open('./data/sanguo\_node.txt','w','utf-8') as f:

f.write("Id Label Weight\r\n")

for name, times in names.items():

f.write(name + ' ' + name + ' ' + str(times) + '\r\n')

with codecs.open('./data/sanguo\_edge.txt', 'w', "utf-8") as f:

f.write("Source Target Weight\r\n")

for name, edges in relationships.items():

for v, w in edges.items():

if w < 50:

f.write(name + ' ' + v + " " + str(w) + " 较少相关"+ "\r\n")

elif w < 100:

f.write(name + ' ' + v + " " + str(w) + " 一般相关"+ "\r\n")

elif w < 250:

f.write(name + ' ' + v + " " + str(w) + " 相关"+ "\r\n")

elif w < 350:

f.write(name + ' ' + v + " " + str(w) + " 很相关"+ "\r\n")

else:

f.write(name + ' ' + v + " " + str(w) + " 密切相关"+ "\r\n")

1. 实现一个简单的数据查询程序

分别基于人名和关系类型的查找.

def searchByOneName(name, relationType):

f = open('./data/sanguo\_edge.txt', encoding='utf-8')

for line in f.readlines():

temp = []

temp = re.split(r' ', line)

if name == temp[0] and relationType in line:

print(line)

print("finsih!!")

def searchByTwoName(name1, name2):

f = open('./data/sanguo\_edge.txt', encoding='utf-8')

for line in f.readlines():

temp = []

temp = re.split(r' ', line)

if name1 == temp[0] and name2 == temp[1]:

print(line)

print("finsih!!")

def searchByrelationType(relationType):

f = open('./data/sanguo\_edge.txt', encoding='utf-8')

for line in f.readlines():

if relationType in line:

print(line)

print("finsih!!")

# 项目总结

因为我自己很喜欢三国演义这部作品，所以也借这个机会了解了以下文本数据分词和信息

提炼相关的知识。总体效果还算满意，但是，自己感觉还有待改善。

如：对刘关张三人联系所得到的评分不高。这显然是与实际存在偏差的。

原因在于，我选取的评价标准是：两个人名同时出现在文本文档一行(在文本文件中差不多是文章一段的篇幅)，记一次。这样处理肯定会有问题，比如实际中诸葛亮和张飞关羽一起登场的频次很高，但是作者描写人物时，主要时诸葛亮和其他势力的交锋，所以得到分数也不是很高.这也是，这种统计方式很大的一个弊病。