**实 验 报 告**

**课程名称** \_\_\_\_\_\_\_\_python程序设计\_\_\_\_\_

**实验项目** \_\_\_\_\_\_\_\_红楼梦分析\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**实验仪器** \_\_\_\_\_\_\_\_\_个人电脑\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**学 院**\_\_\_\_\_\_\_\_计算机学院\_\_\_\_\_\_\_\_\_\_

**专 业**\_\_\_\_\_\_\_\_\_计算机大类\_\_\_\_\_\_\_\_\_

**班级/学号\_\_\_\_计类1805/2018011252\_\_\_**

**学生姓名** \_\_\_\_\_\_\_刘科宏\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**实验日期** \_\_\_\_\_\_\_5月12日\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**成 绩** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**指导教师** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **实验目的**

通过本实验，掌握python语言的基本数据类型列表、字典的使用和操作，以及文件的读写操作，并用python语言完成实际任务。

1. **实验内容**

通过使用python结巴库以及列表字典的使用，提取出红楼梦中的各关键词以及个关键词的出现频率，通过比较前八十回和后四十回的词语出现频率，来得出前八十回用词特点与后四十回的用词特点的相似程度，若相似程度大于等于80%，就可以初步判断前八十回与后四十回的用词特点基本相似，若小于80%，便认为用词特点不是很相似，从而得出不是一个人写的。

1. **实验原理**

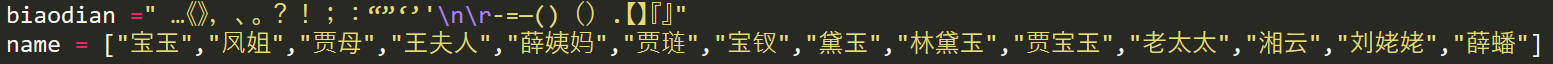
每个人在写作或是说话时都有着自己的用词特点，因此我们可以通过这个特点来初步判断文章是否为一个人所作，而python的结巴库可以很方便的提取出文章的词语，相比对于比较红楼梦前八十回与后四十回，基数很大，因此我们通过利用python对词语的比较便可以较为轻松比较出文章的用词特点不同。

1. **实验步骤**

首先得打开我们必须打开在这段代码中加入jieba库，并且打开红楼梦的全文本文件（如图一）

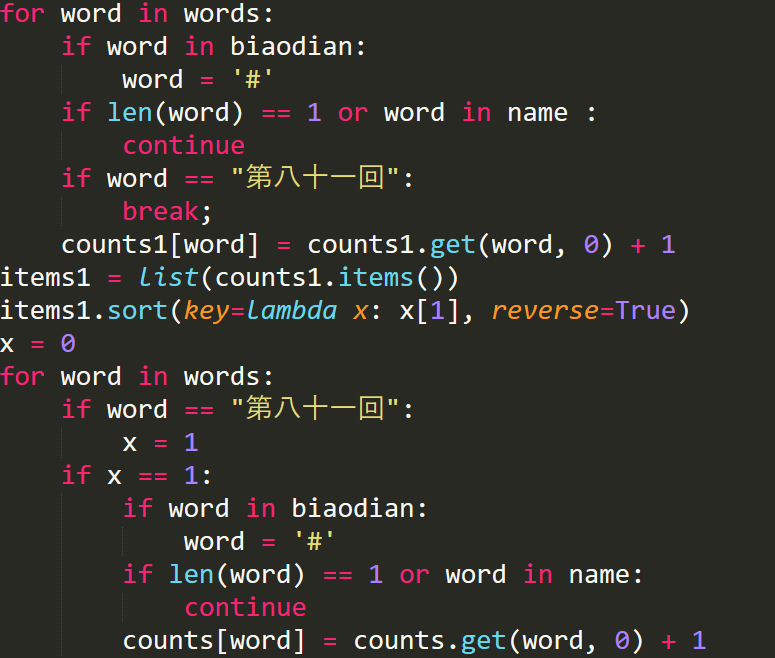
图表 1

然后就可以直接对于文章使用结巴库分词了。

接着我们需要防止把文中的姓名以及标点符号给统计，因此先把它们给写入一个列表当中（如图二）

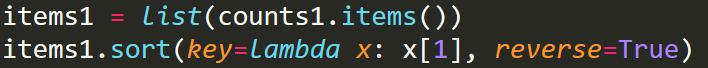
图表 2

准备工作完成之后，接下来就要排数一些使用结巴库分词产生的words中的无用的词语，就比如把标点化为#防止统计，把长度为1的词语全部忽略，因为长度为1的词语若统计会对实验结果造成较大影响，会有许多无意义的单字词语干扰实验结果，尽管会有一些可以表达作者的用词特点。剩下的词语全部get到早先开好的字典当中，又是由于我们进行的是前八十回与后四十回的内容比较，所以这里先统计前八十回的字典，接着使用同样的方法统计后四十回的字典。（如图三）

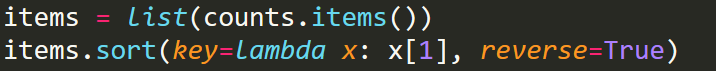


图表 3

这里统计后四十回时先判断word是否已到第八十一回，若是的话让x=1，再开始进行get后四十回的字典。然后我们对两个字典的词频按从到小的顺序进行排序，使用sort函数（如图四图五）



图表 4



图表 5

然后就开始输出了，首先针对于前八十回的字典，按词频大小开始一个个统计输出，每到一个词语时，我们就在后四十回的字典当中找出这个词语，把它以及它在后四十回的词频也同样输出，并比较前八十回中和后四十回的词频关系，由于八十是四十的两倍，所以在这里我判断前八十回同一个词的词频必须得是后四十回的词频的1.5倍到2.5倍之间，否则认定为不正常，若不正常的话则进行一次统计，最后在循环完词频大于100的前八十回词语后，用不正常的个数除以总数，得到这一次的不相似率。（如图六）



图表 6

为了追求更大的准确率，我又一次通过针对于后四十回，以它为主来统计不相似率，方法于上面相似。

在最后分别用1减去两次的不相似率，然后相加除以二，得出平均值，这平均值便就是相似用词特点的相似程度。

其实在中途对于数据的处理想利用到每一回用词频率的方差来判断大概哪些词语是作者的习惯用词，但如果这么做的话，会使循环过多，代码繁杂，不会处理，便使用了这个较为粗略的方法。

1. **实验心得与体会**

这次实验利用Python解决了一个较为困难的问题，使得对Python有了更加深刻的理解以及认识，Python作为一个现在较为流行的语言有着它自己的独特特点，首先是在代码的书写上变得更加简单方便，相较于其他所学语言有很多便捷，可以更加符合人的思路去设计一个实验，对于各个类型的数据处理也有着极大的便利，这一点优点让我惊讶于Python的完整。在其次Python具有着强大的各类库，每个库中的函数都又不断的提现着Python的便捷与简约，就拿这次实验的jieba库来说，或许这个库在其他语言也有，但是使用起来绝对不会比在Python上使用来简单，简单一段代码遍可以轻松把全文化为一个个词语的字符数组，这一就可以通过Python它那简便的字符处理来一个个进行筛选与存放进入字典，这极大的简化了代码的复杂程度。

Python实验对于数据的处理在这一次的实验当中表现得淋漓尽致，就比如Python可以简单的记录每一个词语以及他的频率，这对于其它语言来写可能会花上好几大行，或者复杂好几倍。Python对于数据的处理的方便使得我更加明白了它的流行，在今后的世界，对于大数据的重视，必定会使得Python更加更加流行，因此掌握好Python是一件多么重要的事情，而且这次的Python实验还通过训练电脑来让他自己回答，从某种意义上这也是人工智能的一部分，对于大量数据的处理之后得出结论，从中学习，这一些使用Python可以变得更加简单，这又加大了我对于Python喜爱。这一次实验由于能力的有限，有许多想做的事却没有做出来，这使我感到惋惜，希望在之后的日子当中可以学好Python，利用它来解决更多问题。

**附录：程序源代码**

import jieba

txt = open("hlm\_utf\_8.txt", "r", encoding='utf-8').read()

words = jieba.lcut(txt)

counts1 = {}

counts = {}

biaodian =" …《》，、。？！；：“”‘’'\n\r-=—()（）.【】『』"

name = ["宝玉","凤姐","贾母","王夫人","薛姨妈","贾琏","宝钗","黛玉","林黛玉","贾宝玉","老太太","湘云","刘姥姥","薛蟠"]

for word in words:

if word in biaodian:

word = '#'

if len(word) == 1 or word in name :

continue

if word == "第八十一回":

break;

counts1[word] = counts1.get(word, 0) + 1

items1 = list(counts1.items())

items1.sort(key=lambda x: x[1], reverse=True)

x = 0

for word in words:

if word == "第八十一回":

x = 1

if x == 1:

if word in biaodian:

word = '#'

if len(word) == 1 or word in name:

continue

counts[word] = counts.get(word, 0) + 1

print("前八十回 后四十回")

items = list(counts.items())

items.sort(key=lambda x: x[1], reverse=True)

i = 0

o = 0

y1 = 0

while(True):

word1, count1 = items1[i]

if(count1 < 100):

break

print("{0:<5}{1:>5}".format(word1, count1),end = ' ')

i = i + 1

j = 0

o = 0

while(True) :

word2, count2 = items[j]

if word1 == word2:

print("{0:<5}{1:>5}".format(word2, count2))

o = 1

break

j = j + 1

if j >= len(items) :

break

if int(count1) / float(count2) > 2.5 or float(count1) / float(count2) < 1.5 :

y1 = y1 + 1

if o == 0:

print("\n",end = '')

y1 = y1 / i

i = 0

y2 = 0

o = 0

print("\n\n\n")

print("后四十回 前八十回")

while(True):

word1, count1 = items[i]

if(count1 < 50):

break

print("{0:<5}{1:>5}".format(word1, count1),end = ' ')

i = i + 1

j = 0

o = 0

while(True) :

word2, count2 = items1[j]

if word1 == word2:

print("{0:<5}{1:>5}".format(word2, count2))

o = 1

break

j = j + 1

if j >= len(items) :

break

if int(count1) / float(count2) > (1) or float(count1) / float(count2) < (2/5) :

y2 = y2 + 1

if o == 0:

print("\n",end = '')

y2 = y2 /i

print(y1,y2)

print("相似率：",((1 - y2)\*100 + (1 - y1)\*100)/2,end = '%')