西安电子科技大学

安全前沿讨论班（I） 课程实验报告

实验名称　　简单文本处理

学　院　网络与信息安全学院　　　　　　班 级 1618019

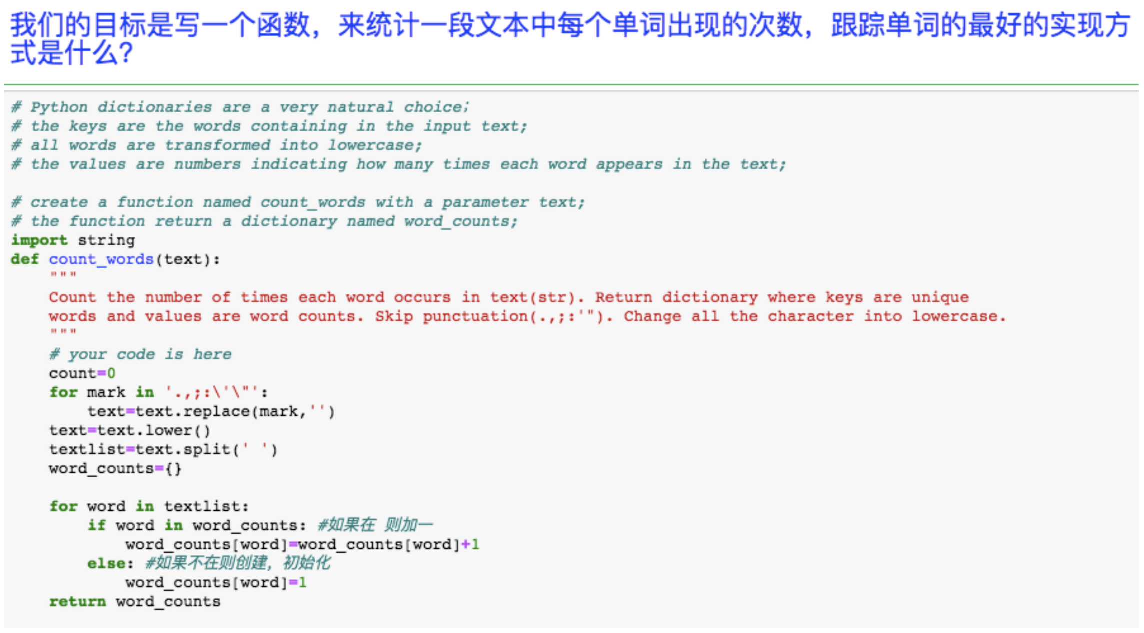
姓　名　曹寅峰　　　　　　　　　　　　学 号　16020610025

实验日期 _____ 年 _____ 月_____日

指导教师评语：   指导教师： ______年____月____日
实验报告内容基本要求及参考格式 一、实验目的 　　应用已经掌握的 Python 程序设计语言的相关知识，对文本进行处理。学习怎样统计文本中的单词频率，怎样通过文件路径获取文件内容，如何利用 pandas 库建立数据表格，以及如何利用 matplotlib 库画图。 二、实验环境 Python 3.5 jupyter

三、实验基本原理及步骤（或方案设计及理论计算）

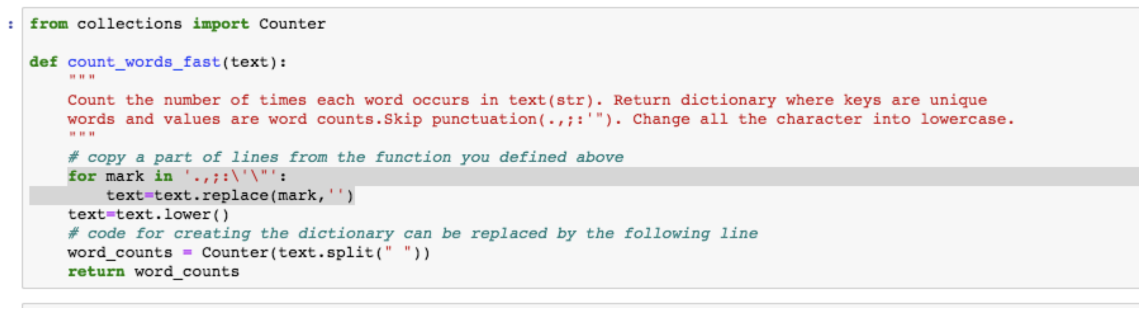
将 2 个压缩包，Books_EngFr.zip 和 Books_GerPort.zip 解压，然后将所有的文件放在同一个名为 Books 的文件夹中。Books 文件夹要和所运行的.ipynb 文件在同一个目录。接下来的实验步骤、要求已经在.ipynb 文件中进行了说明，请按照要求完成。



我们的目标是写一个函数，来统计一段文本中每个单词出现的次数，跟踪单词的最好的实现方式是什么?

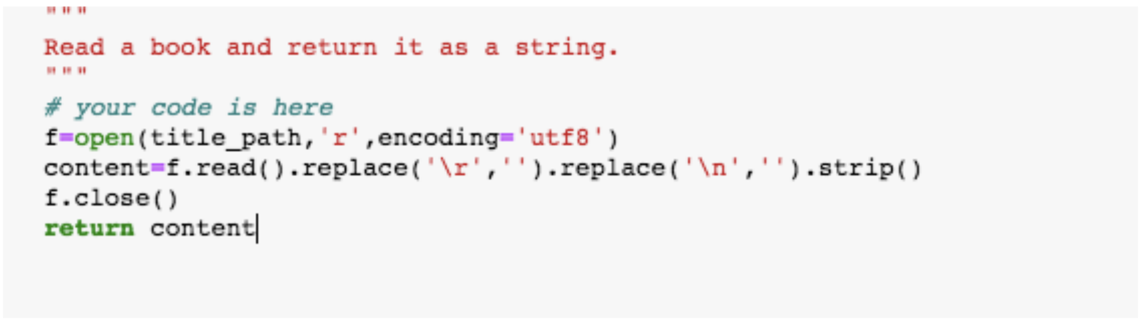
```python
# Python dictionaries are a very natural choice;
# the keys are the words containing in the input text;
# all words are transformed into lowercase;
# the values are numbers indicating how many times each word appears in the text;

# create a function named count_words with a parameter text;
# the function return a dictionary named word_counts;
import string
def count_words(text):
    """
    Count the number of times each word occurs in text(str). Return dictionary where keys are unique
    words and values are word counts. Skip punctuation(.,;:'"). Change all the character into lowercase.
    """
    # your code is here
    count=0
    for mark in '.,;:\'\"':
        text=text.replace(mark,'')
    text=text.lower()
    textlist=text.split(' ')
    word_counts={}

    for word in textlist:
        if word in word_counts: #如果在 则加一
            word_counts[word]=word_counts[word]+1
        else: #如果不在则创建，初始化
            word_counts[word]=1
    return word_counts
```

首先可以用传统方法统计频率，创建一个结果字典，如果一个键不存在则创建它；存在的话则计数加一



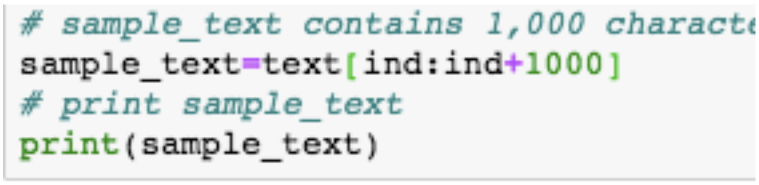
```python
from collections import Counter

def count_words_fast(text):
    """
    Count the number of times each word occurs in text(str). Return dictionary where keys are unique
    words and values are word counts.Skip punctuation(.,;:'"). Change all the character into lowercase.
    """
    # copy a part of lines from the function you defined above
    for mark in '.,;:\'\"':
        text=text.replace(mark,'')
    text=text.lower()
    # code for creating the dictionary can be replaced by the following line
    word_counts = Counter(text.split(" "))
    return word_counts
```

也有比较方便的方法，利用 collections 里面的 counter 类，可以自动根据列表创建计数字典



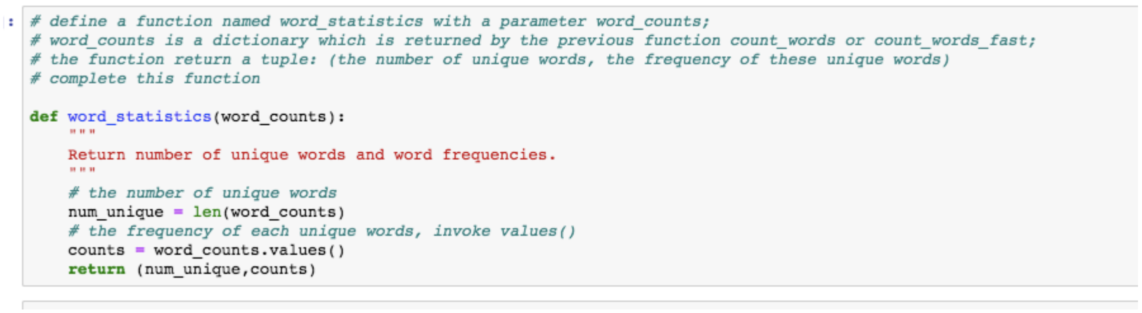
```
    """
    Read a book and return it as a string.
    """
    # your code is here
    f=open(title_path,'r',encoding='utf8')
    content=f.read().replace('\r','').replace('\n','').strip()
    f.close()
    return content
```

通过 open 可以打开文件获取文件描述符，同时去掉里面的换行回车符，最后关闭文件



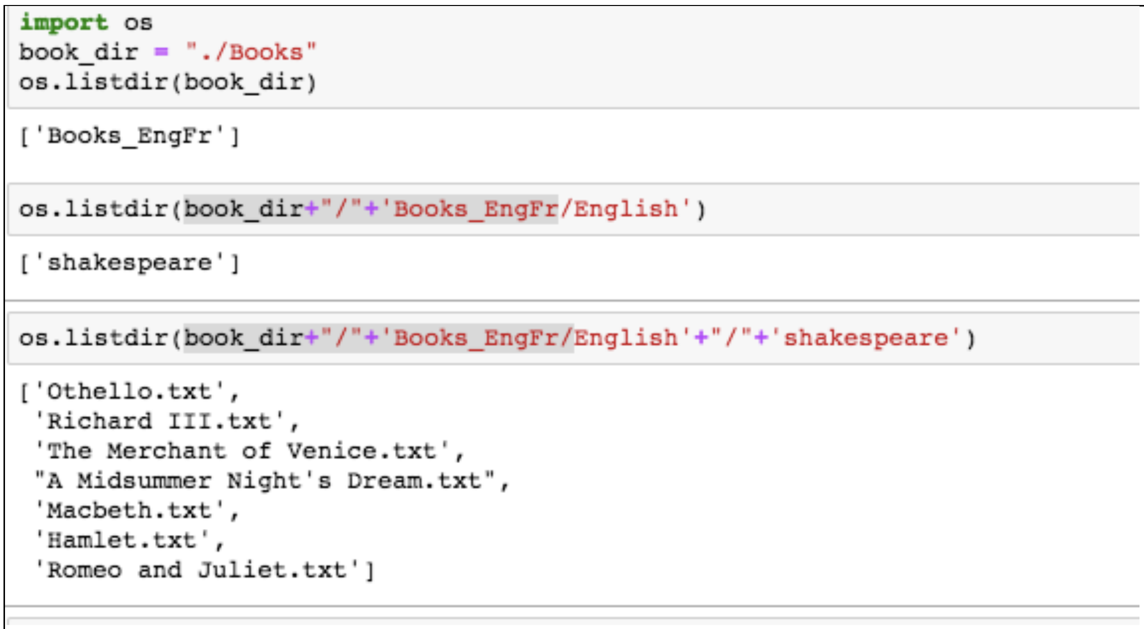
```
# sample_text contains 1,000 characte
sample_text=text[ind:ind+1000]
# print sample_text
print(sample_text)
```

利用切片可以快速截取想要的部分字符串



```
# define a function named word_statistics with a parameter word_counts;
# word_counts is a dictionary which is returned by the previous function count_words or count_words_fast;
# the function return a tuple: (the number of unique words, the frequency of these unique words)
# complete this function

def word_statistics(word_counts):
    """
    Return number of unique words and word frequencies.
    """
    # the number of unique words
    num_unique = len(word_counts)
    # the frequency of each unique words, invoke values()
    counts = word_counts.values()
    return (num_unique,counts)
```

统计字典里的唯一单词和单词总数，可以分别对字典求长度和利用.values()方法获取数量列表。



```
import os
book_dir = "./Books"
os.listdir(book_dir)
```

```
['Books_EngFr']
```

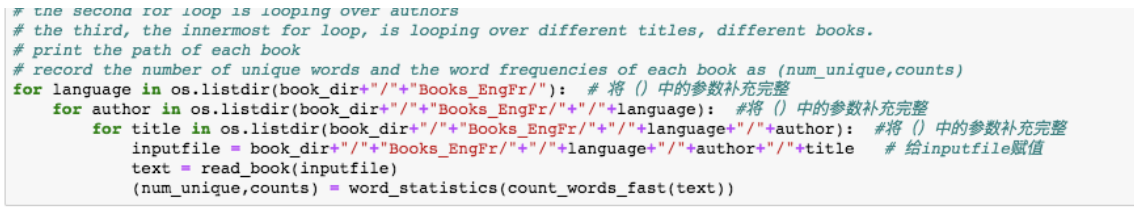
```
os.listdir(book_dir+"/"+'Books_EngFr/English')
```

```
['shakespeare']
```

```
os.listdir(book_dir+"/"+'Books_EngFr/English'+"/"+'shakespeare')
```

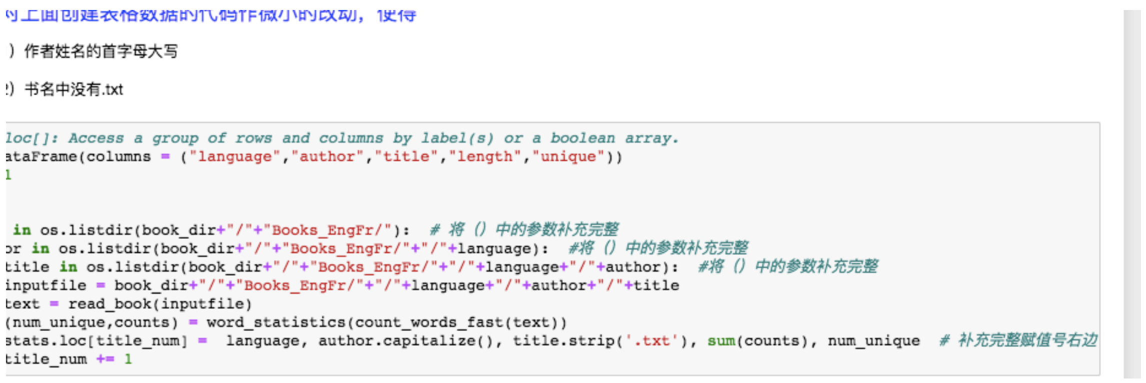
```
['Othello.txt',
 'Richard III.txt',
 'The Merchant of Venice.txt',
 "A Midsummer Night's Dream.txt",
 'Macbeth.txt',
 'Hamlet.txt',
 'Romeo and Juliet.txt']
```

获取路径注意自己的实际路径就可以了。



```
# the second for loop is looping over authors
# the third, the innermost for loop, is looping over different titles, different books.
# print the path of each book
# record the number of unique words and the word frequencies of each book as (num_unique,counts)
for language in os.listdir(book_dir+"/"+"Books_EngFr/"):  # 将 () 中的参数补充完整
    for author in os.listdir(book_dir+"/"+"Books_EngFr/"+"/"+language):  #将 () 中的参数补充完整
        for title in os.listdir(book_dir+"/"+"Books_EngFr/"+"/"+language+"/"+author):  #将 () 中的参数补充完整
            inputfile = book_dir+"/"+"Books_EngFr/"+"/"+language+"/"+author+"/"+title   # 给inputfile赋值
            text = read_book(inputfile)
            (num_unique,counts) = word_statistics(count_words_fast(text))
```

这里注意么一层循环的意思，分别获取语言，作者，标题，然后返回



上面创建表格数据的代码作微小的改动，使得

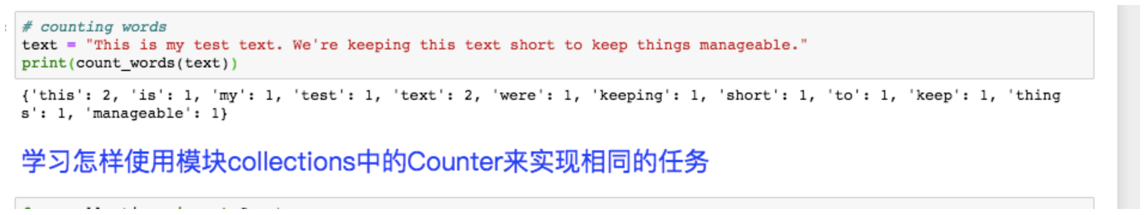
）作者姓名的首字母大写

)）书名中没有.txt

```
loc[]: Access a group of rows and columns by label(s) or a boolean array.
ataFrame(columns = ("language","author","title","length","unique"))
1


 in os.listdir(book_dir+"/"+"Books_EngFr/"):  # 将 () 中的参数补充完整
or in os.listdir(book_dir+"/"+"Books_EngFr/"+"/"+language):  #将 () 中的参数补充完整
title in os.listdir(book_dir+"/"+"Books_EngFr/"+"/"+language+"/"+author):  #将 () 中的参数补充完整
inputfile = book_dir+"/"+"Books_EngFr/"+"/"+language+"/"+author+"/"+title
text = read_book(inputfile)
(num_unique,counts) = word_statistics(count_words_fast(text))
stats.loc[title_num] =  language, author.capitalize(), title.strip('.txt'), sum(counts), num_unique  # 补充完整赋值号右边
title_num += 1
```

对其中的某些元素处理，利用 strip 去掉制定字符串，利用 capitalize()转换为首字母大写格式

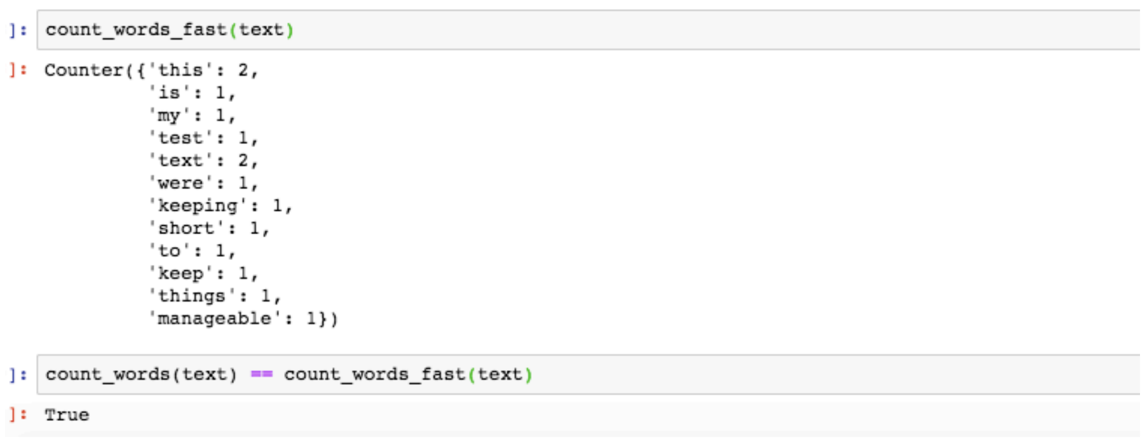
四、实验结果分析及回答问题（或测试环境及测试结果）



```
# counting words
text = "This is my test text. We're keeping this text short to keep things manageable."
print(count_words(text))
```

```
{'this': 2, 'is': 1, 'my': 1, 'test': 1, 'text': 2, 'were': 1, 'keeping': 1, 'short': 1, 'to': 1, 'keep': 1, 'thing
s': 1, 'manageable': 1}
```

学习怎样使用模块collections中的Counter来实现相同的任务



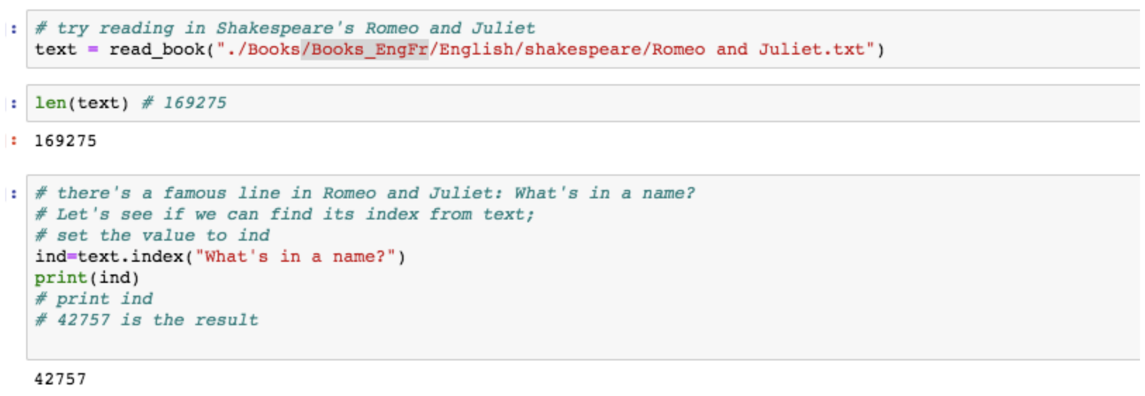
```
]: count_words_fast(text)
```

```
]: Counter({'this': 2,
            'is': 1,
            'my': 1,
            'test': 1,
            'text': 2,
            'were': 1,
            'keeping': 1,
            'short': 1,
            'to': 1,
            'keep': 1,
            'things': 1,
            'manageable': 1})
```

```
]: count_words(text) == count_words_fast(text)
```

```
]: True
```

对比发现两次统计结果一致



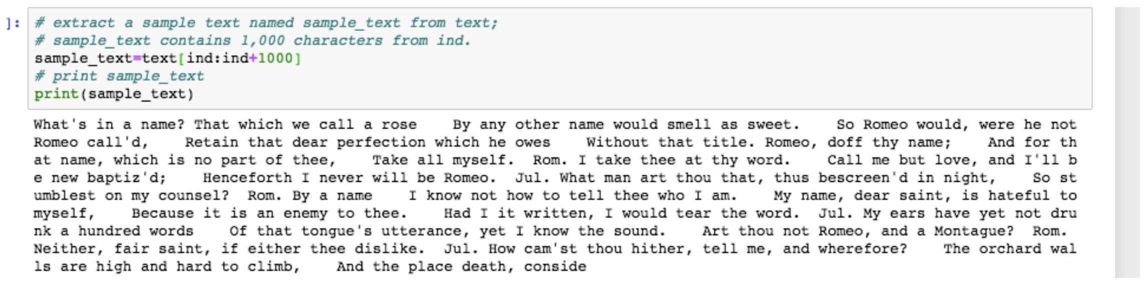
```
# try reading in Shakespeare's Romeo and Juliet
text = read_book("./Books/Books_EngFr/English/shakespeare/Romeo and Juliet.txt")
```

```
len(text) # 169275
```

```
169275
```

```
# there's a famous line in Romeo and Juliet: What's in a name?
# Let's see if we can find its index from text;
# set the value to ind
ind=text.index("What's in a name?")
print(ind)
# print ind
# 42757 is the result
```

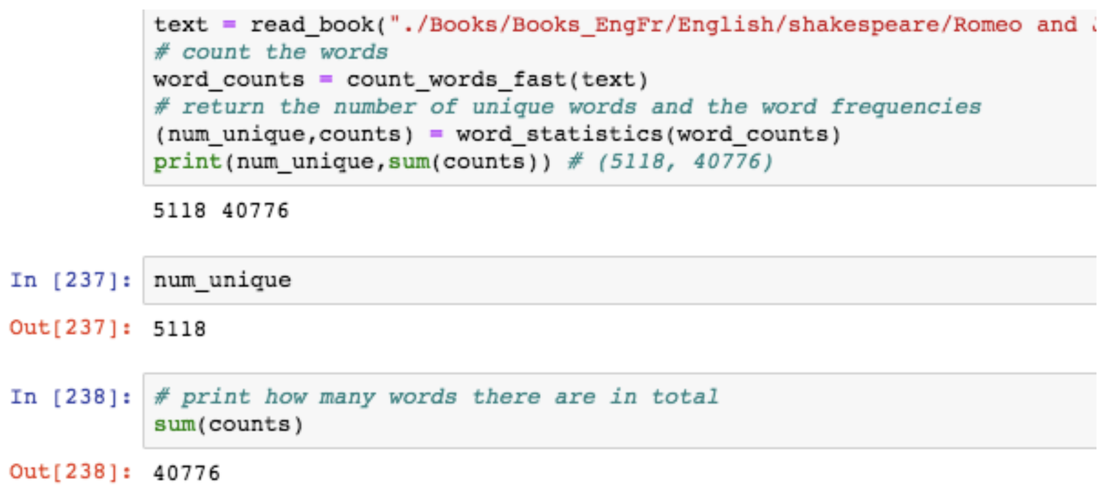
```
42757
```



```
# extract a sample text named sample_text from text;
# sample_text contains 1,000 characters from ind.
sample_text=text[ind:ind+1000]
# print sample_text
print(sample_text)
```

```
What's in a name? That which we call a rose    By any other name would smell as sweet.    So Romeo would, were he not
Romeo call'd,    Retain that dear perfection which he owes    Without that title. Romeo, doff thy name;    And for th
at name, which is no part of thee,    Take all myself.  Rom. I take thee at thy word.    Call me but love, and I'll b
e new baptiz'd;    Henceforth I never will be Romeo.  Jul. What man art thou that, thus bescreen'd in night,    So st
umblest on my counsel?  Rom. By a name    I know not how to tell thee who I am.    My name, dear saint, is hateful to
myself,    Because it is an enemy to thee.    Had I it written, I would tear the word.  Jul. My ears have yet not dru
nk a hundred words    Of that tongue's utterance, yet I know the sound.    Art thou not Romeo, and a Montague?  Rom.
Neither, fair saint, if either thee dislike.  Jul. How cam'st thou hither, tell me, and wherefore?    The orchard wal
ls are high and hard to climb,    And the place death, conside
```

得出了正确的字母数量与位置

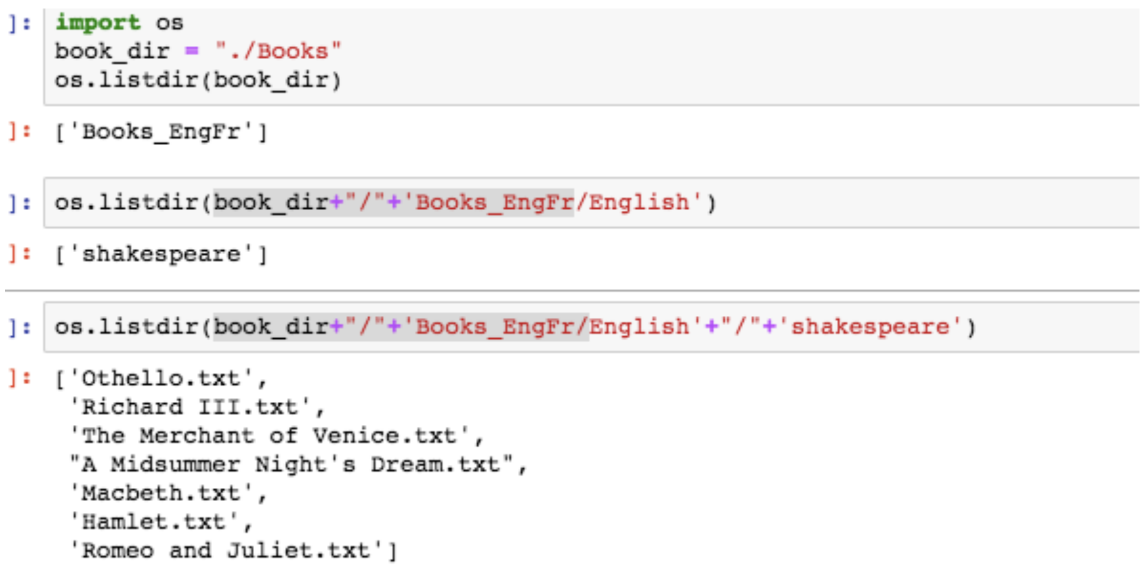


```
text = read_book("./Books/Books_EngFr/English/shakespeare/Romeo and
# count the words
word_counts = count_words_fast(text)
# return the number of unique words and the word frequencies
(num_unique,counts) = word_statistics(word_counts)
print(num_unique,sum(counts)) # (5118, 40776)
```

```
5118 40776
```

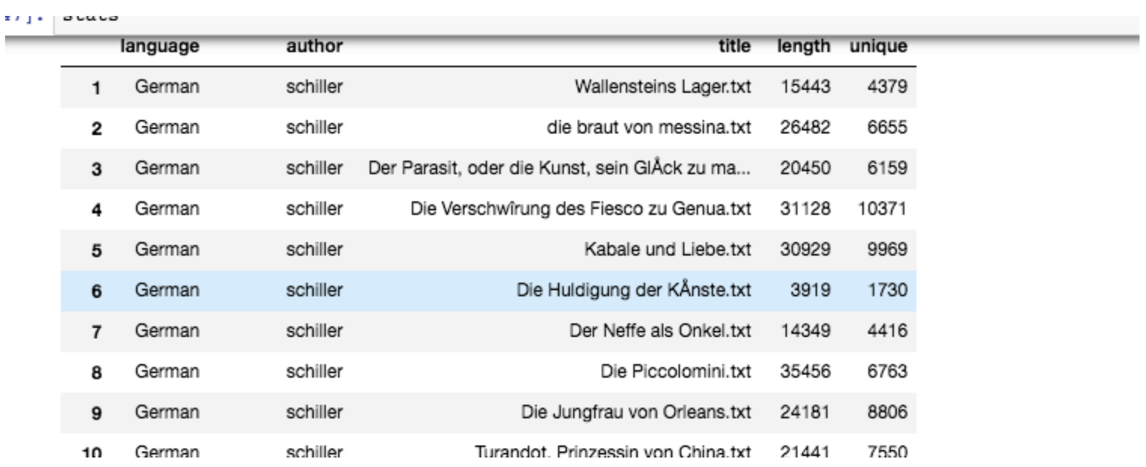
```
In [237]: num_unique

Out[237]: 5118

In [238]: # print how many words there are in total
          sum(counts)

Out[238]: 40776
```

唯一的单词与总数也是正确的



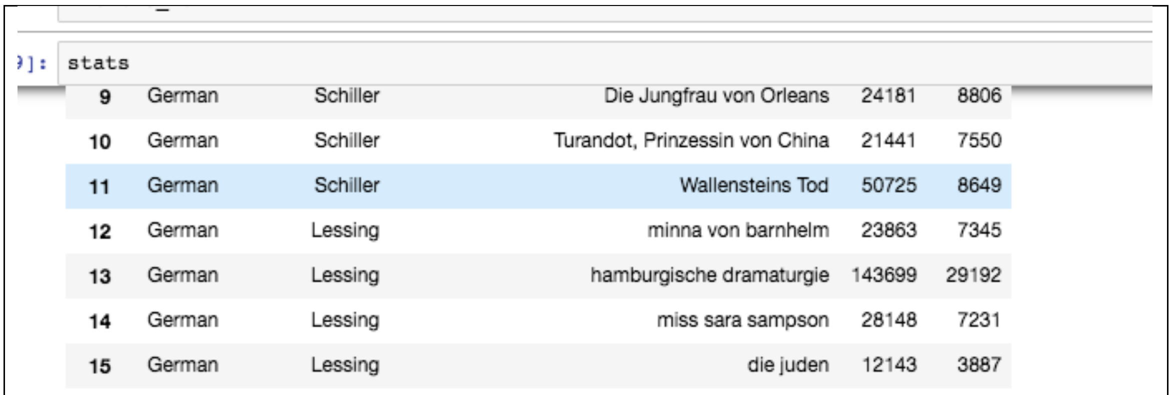
```
]: import os
   book_dir = "./Books"
   os.listdir(book_dir)

]: ['Books_EngFr']

]: os.listdir(book_dir+"/"+'Books_EngFr/English')

]: ['shakespeare']

]: os.listdir(book_dir+"/"+'Books_EngFr/English'+"/"+'shakespeare')

]: ['Othello.txt',
    'Richard III.txt',
    'The Merchant of Venice.txt',
    "A Midsummer Night's Dream.txt",
    'Macbeth.txt',
    'Hamlet.txt',
    'Romeo and Juliet.txt']
```

路径获取正确



	language	author	title	length	unique
1	German	schiller	Wallensteins Lager.txt	15443	4379
2	German	schiller	die braut von messina.txt	26482	6655
3	German	schiller	Der Parasit, oder die Kunst, sein GlÅck zu ma...	20450	6159
4	German	schiller	Die Verschwîrung des Fiesco zu Genua.txt	31128	10371
5	German	schiller	Kabale und Liebe.txt	30929	9969
6	German	schiller	Die Huldigung der KÅnste.txt	3919	1730
7	German	schiller	Der Neffe als Onkel.txt	14349	4416
8	German	schiller	Die Piccolomini.txt	35456	6763
9	German	schiller	Die Jungfrau von Orleans.txt	24181	8806
10	German	schiller	Turandot, Prinzessin von China.txt	21441	7550

利用 pd 库进行多维统计

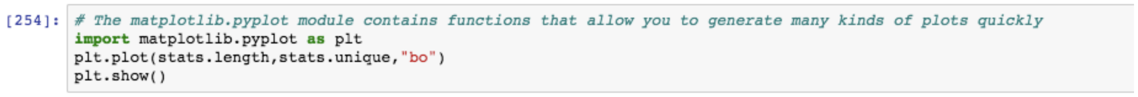


```
]: stats
```

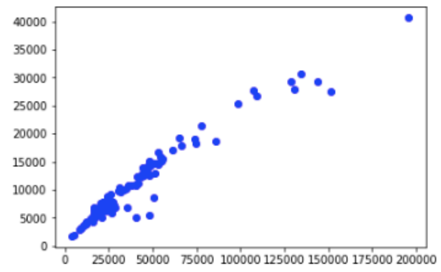
9	German	Schiller	Die Jungfrau von Orleans	24181	8806
10	German	Schiller	Turandot, Prinzessin von China	21441	7550
11	German	Schiller	Wallensteins Tod	50725	8649
12	German	Lessing	minna von barnhelm	23863	7345
13	German	Lessing	hamburgische dramaturgie	143699	29192
14	German	Lessing	miss sara sampson	28148	7231
15	German	Lessing	die juden	12143	3887

首字母大写，去掉 txt

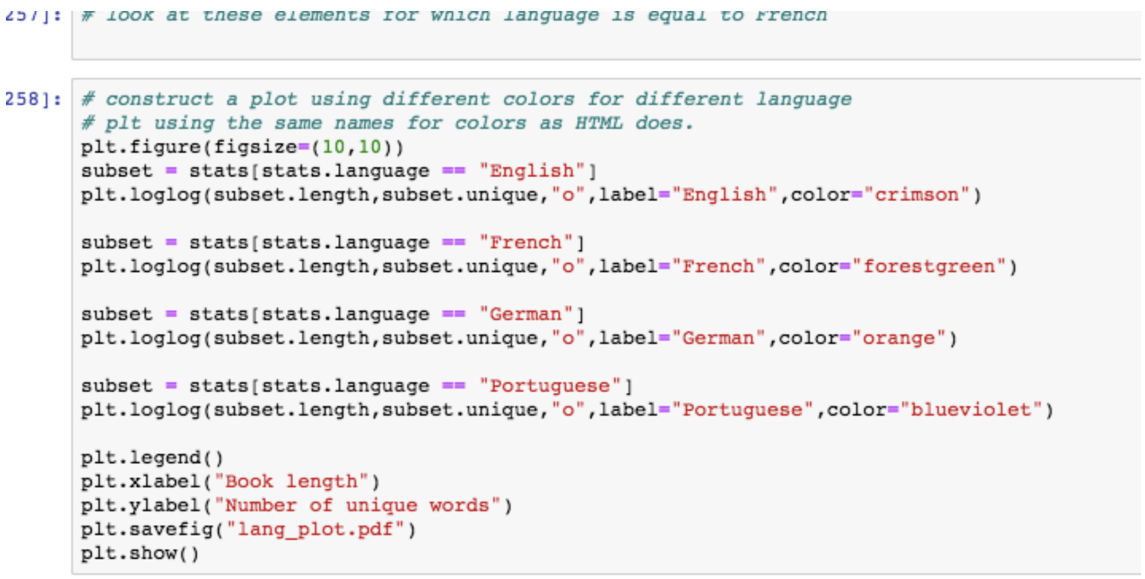
学习怎样使用matplotlib.pyplot画出书籍的长度和单词频率统计图。



```
[254]: # The matplotlib.pyplot module contains functions that allow you to generate many kinds of plots quickly
import matplotlib.pyplot as plt
plt.plot(stats.length,stats.unique,"bo")
plt.show()
```

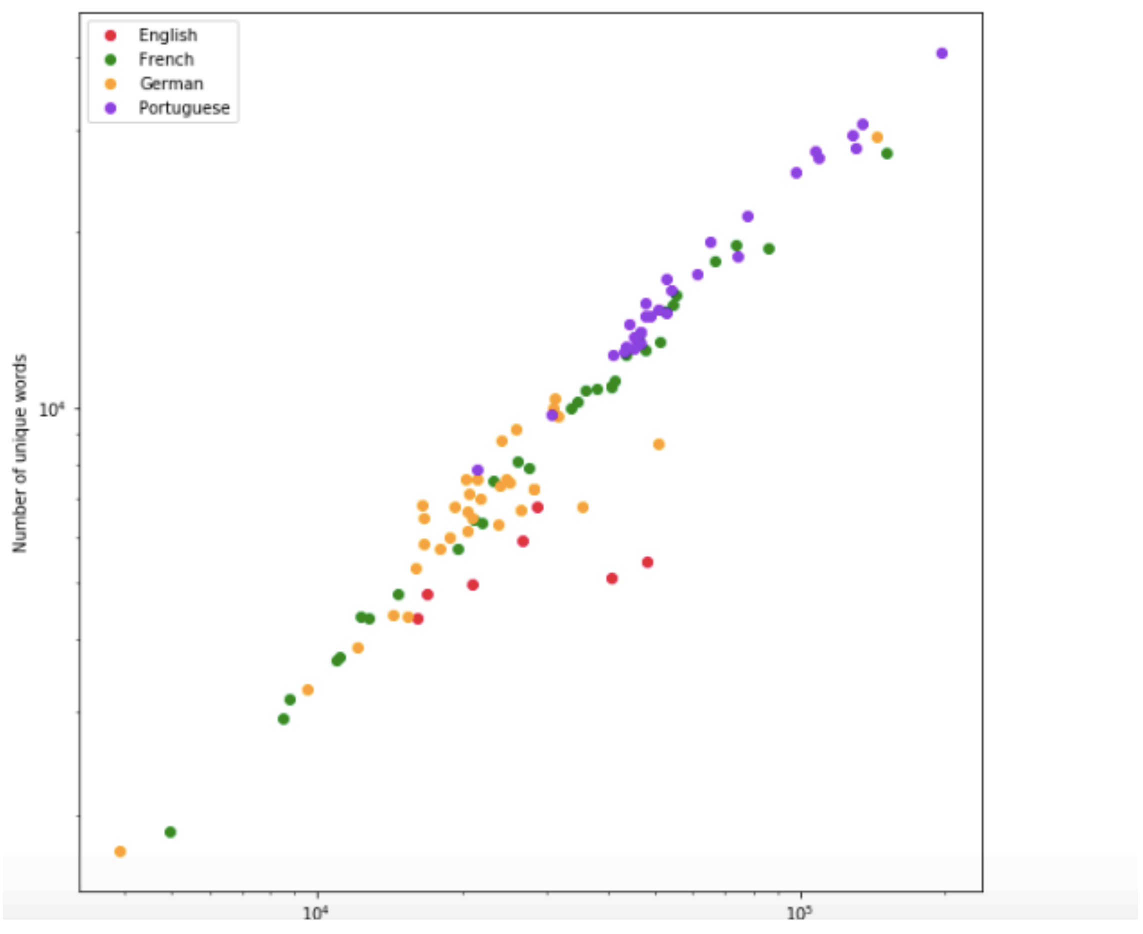


利用 mplot 库画图，制定 x 坐标为文本长度，y 坐标为唯一单词数量，bo 代表蓝色实心点



```
257]: # look at these elements for which language is equal to French

258]: # construct a plot using different colors for different language
# plt using the same names for colors as HTML does.
plt.figure(figsize=(10,10))
subset = stats[stats.language == "English"]
plt.loglog(subset.length,subset.unique,"o",label="English",color="crimson")

subset = stats[stats.language == "French"]
plt.loglog(subset.length,subset.unique,"o",label="French",color="forestgreen")

subset = stats[stats.language == "German"]
plt.loglog(subset.length,subset.unique,"o",label="German",color="orange")

subset = stats[stats.language == "Portuguese"]
plt.loglog(subset.length,subset.unique,"o",label="Portuguese",color="blueviolet")

plt.legend()
plt.xlabel("Book length")
plt.ylabel("Number of unique words")
plt.savefig("lang_plot.pdf")
plt.show()
```

同理，可以同时画出多个情况同时存在的对比图，从而得出统计规律


English
French
German
Portuguese
Number of unique words
10⁴
10⁴
10⁵