

西南交通大学



互联网搜索引擎课程设计报告

学号： 2015114716

班级： 计算机 2015-03 班

学生： 史晨阳

指导老师： 吴晓

目录

1. 项目要求.....	3
2. 项目设计与结果展示.....	3
2.1 编程环境说明.....	3
2.2 网页爬虫设计.....	4
2.2.1 英文网页爬取.....	4
2.2.2 中文网页爬取.....	8
2.3 英文文本处理.....	11
2.3.1 工具包安装.....	11
2.3.2 英文单词字符化.....	11
2.3.3 去除停用词.....	12
2.3.4 抽取单词词干.....	13
2.4 中文文本处理.....	15
2.4.1 工具包安装.....	15
2.4.2 中文文本分词.....	16
2.4.3 构造中文停用词表.....	17
2.4.4 去除中文停用词.....	18
3. 参考网页.....	20
4. 附录(整个工程源代码).....	20

项目一.搜索引擎文本预处理

1. 项目要求

通过下载引擎(Web Crawler/Spider)自动下载至少 500 个英文文档/网页以及 50 个中文文档/网页。并保留原始的文档/网页备份(如: News_1_Org.txt)。

编程对所下载的文档进行自动预处理:

- 1.将各个单词进行字符化, 完成删除特殊字符等操作。
- 2.调研并选择合适的中文分词技术和工具实现中文分词。可参考《Lucene 分词器比较》
- 3.删除英文停用词(Stop Word)
- 4.删除中文停用词
- 5.调用或者编程实现 Porter Stemming 功能
- 6.将中文文档进行字符化, 即可被搜索引擎索引的字符单元
- 7.对于英文文档, 经过以上处理之后, 将经过处理之后所形成简化文档保存(如: News_1_E.txt), 以备以后的索引处理。
- 8.对于中文文档, 经过以上处理之后, 将经过处理之后所形成简化文档保存(如: News_1_C.txt), 以备以后的索引处理。

注意:

尽可能将所有步骤集成为一个完整的自动化系统。如无法, 则需编程实现各个步骤(模块), 然后利用中间文件传递结果。

下载引擎若自己编写那么有加分。

尽量下载处理更多的网页, 以备后期搜索使用。

可利用提供的停用词表, 或自己根据不同应用所生成的停用词表。

2. 项目设计与结果展示

2.1 编程环境说明

目前爬虫和文本处理用的最多而且最方便的是 python, 因此整个项目用 python 语言编写, 基于 Anaconda2 的 Jupyter Notebook。

Jupyter Notebook (原 IPython Notebook), 是一个开源的、交互式的科学计算 Web 应用, 它支持实时的代码执行, 支持文本内容嵌入, 包括数据公式、数据可视化图表、视频等, 通常被用于数据清洗和转换、数值拟合、统计建模、机器学习等, 是数据科学相关研发人员的得力工具。简单来讲, Jupyter Notebook 可以执行并保存代码的执行结果, 下一次打开时无需重新运行即可查看, 是一种可以实现可读性分析的灵活工具。Jupyter Notebook 具有以下几个优势: (1) 支持 40 多种程序语言, 包括数据科学分析中常用的 Python、R、Julia 和 Scala;

(2) 可以通过邮件、Dropbox、GitHub 以及 nbviewer (Jupyter Notebook 查看器, 用于渲染 .ipynb 文件, 并支持 HTML 和 PDF 输出) 共享 Notebook 文件;
(3) 交互式窗口, 代码输出结果可以是图片、视频、LaTeX、JavaScript 等, 有助于数据的实时处理及可视化展示。(4) 支持多种类似 Apache Spark 的大数据处理工具以及相关库, 如 pandas、scikit-learn、ggplot2 等。

2.2 网页爬虫设计

2.2.1 英文网页爬取

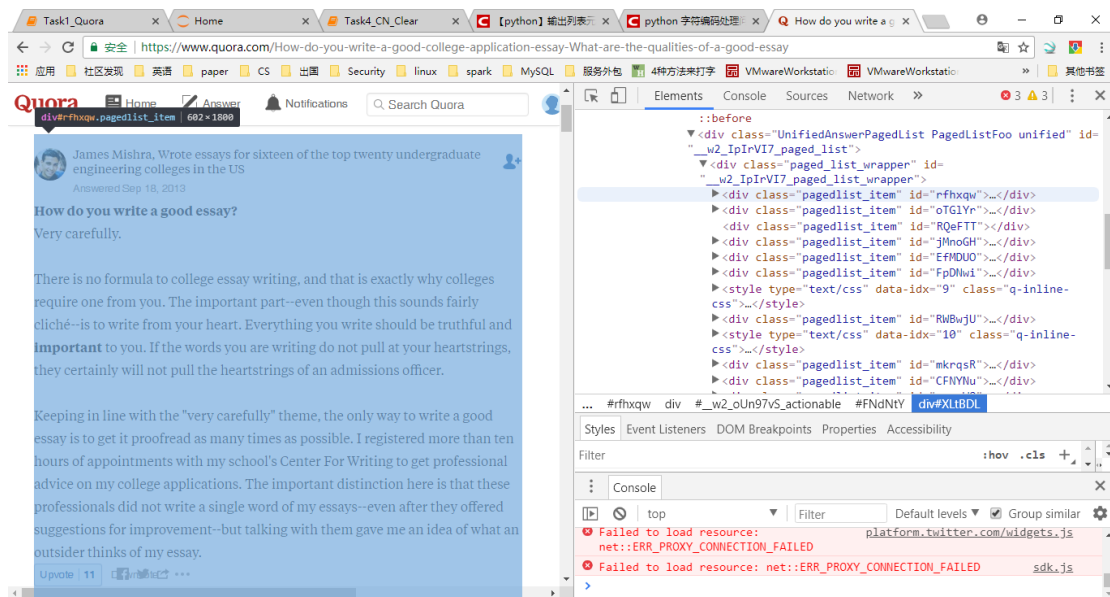
这里我用的是 python 的 BeautifulSoup 来分析和抓取网页的 html 结构。

前 236 个文档都是爬取的网站 <http://csrankings.org/> 的内容。

后面 237-500 爬取的是 Quora 问题下面的回答

<https://www.quora.com/How-do-I-write-the-best-college-application-essay>。

我们首先看 Quora 页面, 我首先找到该问题下面的回答模块所在的位置, 在一个 `div class = pagedlist_warpper` 下面每一条回答对应一个 `div class = pagedlist_item`, 所以爬下来这些所有 `div class = pagedlist_item` 里面的文本即可。



代码如下, 文本内容都在一个 table 的 List 里

```
In [5]: import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import urllib2
import warnings
warnings.filterwarnings("ignore")

web = 'https://www.quora.com/How-do-I-write-the-best-college-application-essay'

req = urllib2.Request(web) #request to visit web
page = urllib2.urlopen(req) #open this url
soup = BeautifulSoup(page, 'lxml') ##将网页源码造成BeautifulSoup对象, 方便操作
table = soup.find_all("div", {"class": "pagedlist_item"}) #找到这个标签
```

比如打印一个回答的文本，和 quora 上的原回答是完全一样的。

79 Answers



Ryan Nelson, Writer at GradLime

Answered Dec 7

A good college application essay does two things:

1. Clearly addresses the prompt.
2. Showcases something unique or interesting about you (usually through a story).

It's like putting a face to your name. The whole point of college application essays is to show the admissions office something they can't learn about you from test scores and data.

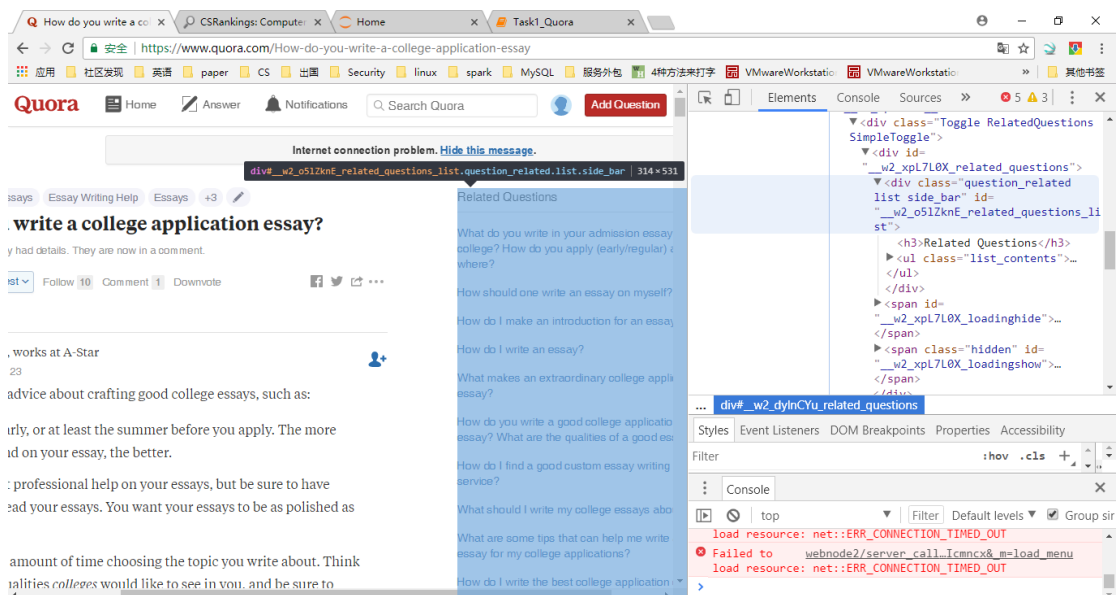
Here are some tips to help you write a strong essay:

Brainstorming: Think of several ways to address the prompt.

```
In [29]: print table[0].get_text()
```

Ryan Nelson, Writer at GradLime Answered Dec 6, 2017 • Author has 53 answers and 47.3k answer views A good college application essay does two things: Clearly addresses the prompt. Showcases something unique or interesting about you (usually through a story). It's like putting a face to your name. The whole point of college application essays is to show the admissions office something they can't learn about you from test scores and data. Here are some tips to help you write a strong essay: Brainstorming: Think of several ways to address the prompt. Your first idea isn't always going to be the best. So don't get invested in a direction until you map out yo

然后在这个页面上爬取右侧——其他问题的链接。



代码如下。但这样得到的 url 不完整。

```
In [23]: URLquestion = []
for row in soup.find_all("li", {"class": "related_question"}):
    for a in row.find_all('a', href=True):
        URLquestion.append(a['href'])
URLquestion

Out[23]: ['/How-do-you-write-a-good-college-application-essay-What-are-the-qualities-of-a-good-essay',
'/How-do-you-write-a-college-application-essay',
'/How-do-I-find-a-good-custom-essay-writing-service',
'/How-do-I-write-college-essays-faster',
'/What-do-you-write-in-your-admission-essay-for-college-How-do-you-apply-early-regular-and-where',
'/What-is-the-importance-of-writing-a-college-application-essay']
```

用下面的代码给所有的 url 添加成完整的链接

```
In [24]: #加上完整的url链接
URLfull = []
for i in URLquestion:
    URLfull.append('https://www.quora.com' + i)
for i in URLfull:
    print i

https://www.quora.com/How-do-you-write-a-good-college-application-essay-What-are-the-qualities-of-a-good-essay
https://www.quora.com/How-do-you-write-a-college-application-essay
https://www.quora.com/How-do-I-find-a-good-custom-essay-writing-service
https://www.quora.com/How-do-I-write-college-essays-faster
https://www.quora.com/What-do-you-write-in-your-admission-essay-for-college-How-do-you-apply-early-regular-and
```

用同样的方式获取问题下面的所有回答，即依次访问这些问题的页面，比如访问第一个 url，同样的方式找到所有 div class = pagedlist_item 的所有文本，放在 table2 里面，再把 table2 的每条内容连在 table 上。当以上所有链接访问之后，再次打印 table 的长度已经有 340。说明已经存放了 340 个回答。

```
In [ ]: for url in URLfull:
        req = urllib2.Request(url) #request to visit web
        page = urllib2.urlopen(req) #open this url
        soup = BeautifulSoup(page, 'lxml')
        table2 = soup.find_all("div", {"class" : "pagedlist_item"})
        for t in table2:
            table.append(t)
```

```
In [66]: for t in table2:
        table.append(t)
```

```
In [67]: len(table)
```

```
Out[67]: 340
```

现在只需要把所有的回答保存成文件即可。

```
In [69]: import sys
        reload(sys)
        sys.setdefaultencoding("utf8")
        num = 237
        for content in table:
            if (len(content) == 0):
                continue;
            with open('News_' + str(num) + '_Org.txt', 'w') as f:
                f.write(content.get_text())
            num += 1
            if (num > 500):
                break;
```

再来看 CSRanking 页面

The screenshot shows the CSRankings website in a browser. The page title is "CSRankings: Computer Science Rankings". It features a navigation bar with links for "All Areas", "Systems", and "Rank Institution". The "Rank Institution" section lists various universities and their faculty members. The browser's developer tools are open, showing the HTML structure of the page. The "Elements" panel highlights a table with columns "Rank" and "Institution". The "Console" panel shows a red error message: "net::ERR_CONNECTION_TIMED_OUT".

首先在页面上找到我要抓取的位置为 Rank Institution 下的各个美国和加拿大的高校在 AI 领域和计算机交叉领域下的教授各个人主页，那么可以看到每一个教授的主页链接都存在一个 `<small>` 的 tag 里，所以找到所有的 `<small>` 标签里的所有 href 存放在一个 List 里，然后依次访问这些链接，抓取个人主页所需要的文本。

由于每一个主页的 html 设计是不同的，所以我统一都抓取的 body 里面的所有文本。

```
In [2]: f = open('URL-homepage.txt', 'r').read()
        print f
        f.close()
```

```
http://vision.stanford.edu/feifeili/
http://nlp.stanford.edu/manning/
https://cs.stanford.edu/~pliang/
http://cvgl.stanford.edu/silvio/
https://cs.stanford.edu/people/jure/
http://web.stanford.edu/~jurafsky/
https://cs.stanford.edu/~ermon/
```

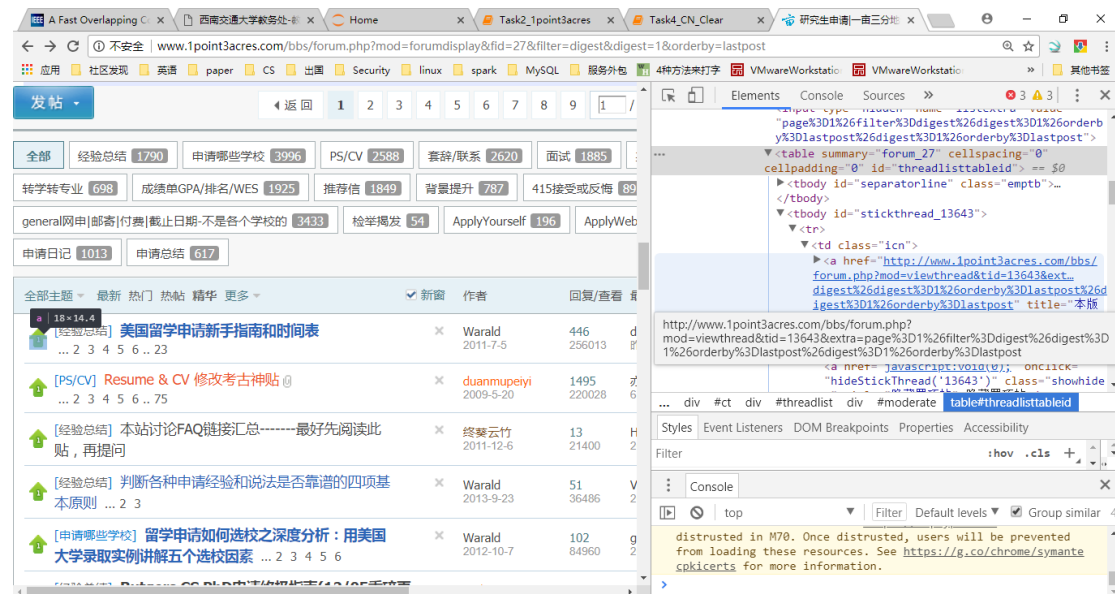
```
num = 1
with open('URL-homepage.txt', 'r') as f: #读取url链接
    f.readline()
    for line in f:
        req = urllib2.Request(line) #request to visit web
        page = urllib2.urlopen(req) #open this url
        soup = BeautifulSoup(page, 'lxml') #将网页源码构造成BeautifulSoup对象，方便操作
        if(soup.body != None):
            text = soup.body #body里的文本
        else:
            text = soup.head

    with open('News_' + str(num) + '_Org.txt', 'w') as f:
        f.write(text.get_text())
        num += 1
        if (num > 251):
            break;
```

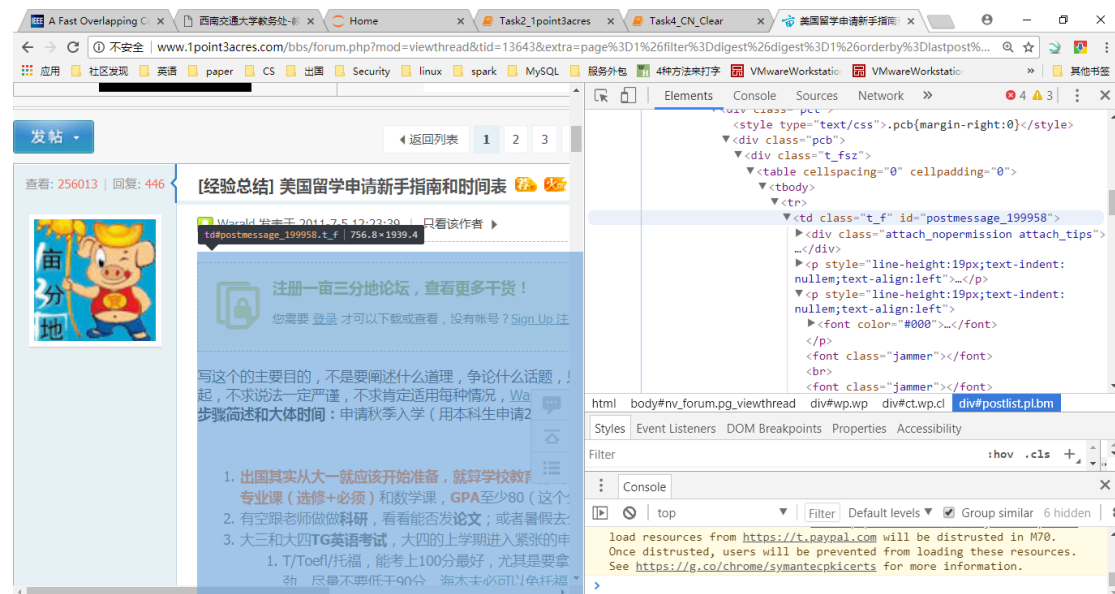
2.2.2 中文网页爬取

<http://www.1point3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&filter=digest&digest=1>

我抓取一亩三分地上的精华帖，方法和上面一样。



由于论坛格式复杂，而且各种 CSS 样式十分难以处理。所以我找的是每一个精华帖子中的楼层 1 的楼主的内容。



首先访问精华帖的第一页

```
In [1]: import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import urllib2
import warnings
warnings.filterwarnings("ignore")
#一亩三分地留学论坛精华帖
web = 'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&filter=digest&digest=1'
req = urllib2.Request(web) #request to visit web
page = urllib2.urlopen(req) #open this url
soup = BeautifulSoup(page, 'lxml') ##将网页源码构造成BeautifulSoup对象, 方便操作
```

这是在本页找到的所有翻页的链接，访问就可以抓取下一页同样的内容。
链接在<div = pg>的 Tag 里

```
In [10]: #import sys
#reload(sys)
#sys.setdefaultencoding("utf8")
#所有页面的url
Page = []
table = soup.find("div", "pg") #找到这个标签
for p in table.find_all('a', href = True):
    Page.append(p['href'])

Page
```

```
Out[10]: ['http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=2',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=3',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=4',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=5',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=6',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=7',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=8',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=9',
'http://www.lpoint3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&digest=1&orderby=lastpost&filter=digest&digest=1&orderby=lastpost&page=2']
```

这些是一个页面中的所有帖子的链接，一页有 50 条帖子。每个帖子的标题在<td class = icn>的标签里，抓取这个 Tag 里面的所有的链接保存在 URL 的 list[]

```
In [26]: import requests
URL = []
for i in soup.find_all('td', class_ = "icn"): #找帖子标题
    for a in i.find_all('a', href = True):
        URL.append(a['href'])
for num in range(0, 9):
    req = urllib2.Request(Page[num]) #翻页
    page = urllib2.urlopen(req) #open current url
    soup = BeautifulSoup(page, 'lxml')
    for i in soup.find_all('td', class_ = "icn"):
        for a in i.find_all('a', href = True):
            URL.append(a['href'])

URL
```

```
Out[26]: ['http://www.lpoint3acres.com/bbs/forum.php?mod=viewthread&tid=13643&extra=page%3D1%26filter%3Ddigest%26digest%3D1%26orderby%3Dlastpost%26digest%3D1%26orderby%3Dlastpost',
'http://www.lpoint3acres.com/bbs/forum.php?mod=viewthread&tid=19&extra=page%3D1%26filter%3Ddigest%26digest%3D1%26orderby%3Dlastpost%26digest%3D1%26orderby%3Dlastpost',
'http://www.lpoint3acres.com/bbs/forum.php?mod=viewthread&tid=19800&extra=page%3D1%26filter%3Ddigest%26digest%3D1%26orderby%3Dlastpost%26digest%3D1%26orderby%3Dlastpost',
'http://www.lpoint3acres.com/bbs/forum.php?mod=viewthread&tid=72476&extra=page%3D1%26filter%3Ddigest%26digest%3D1%26orderby%3Dlastpost%26digest%3D1%26orderby%3Dlastpost',
```

每一个 URL 都是一个精华帖链接。例如我们现在打印一个 URL 中楼 1 的内容:

```
In [13]: import requests
text = requests.get('http://www.lpoint3acres.com/bbs/forum.php?mod=viewthread&tid=13643&extra=page%3D1%26filter%3
text = BeautifulSoup(text, from_encoding="gb18030")

In [14]: print text.prettify()

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transit
ional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<title>
    美国留学申请新手指南和时间表【一亩三分地论坛研究生申请版】 - Powered by Discuz!
</title>
<link href="http://www.lpoint3acres.com/bbs/favicon.ico" rel="Bookmark"/>
<link href="http://www.lpoint3acres.com/bbs/favicon.ico" rel="shortcut icon"/>
<link href="http://www.lpoint3acres.com/bbs/thread-13643-1-1.html" rel="canonical"/>
<meta content="美国留学, 美国留学申请, 时间表" name="keywords"/>
<meta content="美国留学申请新手指南和时间表" name="description"/>
<meta content="Discuz! X3" name="generator"/>
<meta content="Discuz! Team and Comsenz UI Team" name="author"/>
```

用 gb18030 编码，先展示 html 结构如上图。然后找到楼主的板块，在 `<td class = t f>` 的标签里面

```
In [15]: # Title: Titles are in h3 tags with an attribute class="event-title".
# Titles are in h3 tags with an attribute class="event-title".
mainTags = text.find_all('td', {'class': "t_f"})
print mainTags[0].get_text()
```

写这个的主要目的，不是要阐述什么道理，争论什么话题，只是想以时间为序，把申请的大体过程和所要做的事情，串在一起，不求说法一定严谨，不求肯定适用每种情况，Wald只求给大家一个可以简单参考的大体步骤和计划表。步骤简述和大体时间：申请秋季入学（用本科生申请2018秋季入学为例）

出国其实从大一就应该开始准备，就算学校教育方式你很不喜欢、任课老师你觉得很肤浅，也得把功课学好，尤其是专业选修（必修+必须）和数学课，GPA至少80（这个分数其实也很低，建议至少考个85），排名不要太高看有空跟老师做做科研，看看能否发论文；或者暑假去国外做实习，体会、考虑自己喜欢做什么大三和四大四英语考试，大四的上学期进入紧张的申请阶段。申请研究生一般需要的英语考试：

T/Toefl/托福，能考上100分最好，尤其是要拿奖学金的，即使能力有限，Wald也建议你使劲使劲再使劲，尽量不要低于90分。海本未必可以免托福，雅思未必能代替托福。G：商学院大多数项目需要GMAT，其他理工科需要GRE[最新版答案]GRE/TOEFL/IBT托福考试多少分够用？某个分数是否太低？一般学校申请deadline截止日期是1/15左右（更早的12/1也有，晚到来年的2-3月也有），如果申请高排名学校的奖学金，尽量不晚于11月解决所有的考试，其他情况下，尽管不要晚于1月；某些情况

遍历 URL[1]，即访问所有的精华帖，然后把如上面显示的文本保存在文件中

```
In [20]: import requests
num = 501
for url in URL:
    text = requests.get(url).text
    text = BeautifulSoup(text, from_encoding="gb18030")
    mainTags = text.find_all('td', {'class': "t_f"})
    with open('News_' + str(num) + '_Org.txt', 'w') as f:
        f.write(mainTags[0].get_text())
    num += 1
```

快速访问	名称	修改日期	类型	大小
此电脑	News_998_Org.txt	2018/4/20 0:03	文本文档	2 KB
Desktop	News_999_C.txt	2018/4/22 20:05	文本文档	5 KB
视频	News_999_Org.txt	2018/4/20 0:03	文本文档	8 KB
图片	News_1000_C.txt	2018/4/22 20:05	文本文档	1 KB
文档	News_1000_Org.txt	2018/4/20 0:03	文本文档	1 KB
下载	News_1001_Org.txt	2018/4/20 0:03	文本文档	1 KB
音乐	News_1002_Org.txt	2018/4/20 0:03	文本文档	9 KB
易 (C:)	News_1003_Org.txt	2018/4/20 0:03	文本文档	7 KB
样 (D:)	News_1004_Org.txt	2018/4/20 0:03	文本文档	1 KB
千 (E:)	News_1005_Org.txt	2018/4/20 0:03	文本文档	1 KB
本地磁盘 (F:)	News_1006_Org.txt	2018/4/20 0:03	文本文档	1 KB
盘 (G:)	News_1007_Org.txt	2018/4/20 0:03	文本文档	3 KB
	News_1008_Org.txt	2018/4/20 0:03	文本文档	1 KB
	News_1009_Org.txt	2018/4/20 0:03	文本文档	1 KB
	News_1010_Org.txt	2018/4/20 0:03	文本文档	2 KB
网络	News_1011_Org.txt	2018/4/20 0:03	文本文档	1 KB
	News_1012_Org.txt	2018/4/20 0:03	文本文档	5 KB
	News_1013_Org.txt	2018/4/20 0:03	文本文档	1 KB
	News_1014_Org.txt	2018/4/20 0:03	文本文档	1 KB

2.3 英文文本处理

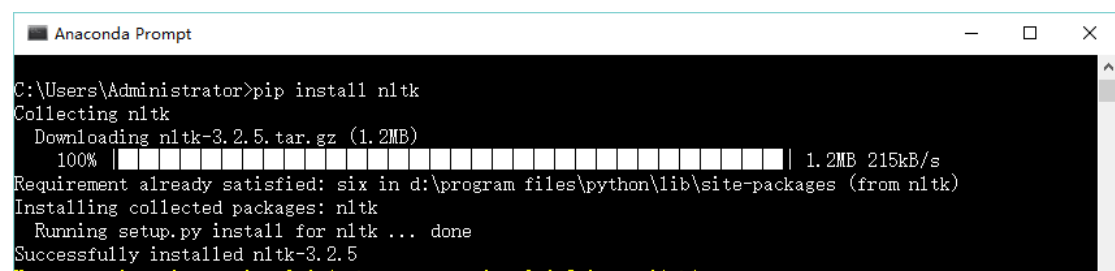
2.3.1 工具包安装

在英文中，单词之间是以空格作为自然分界符组成语句，语句之间再利用标点分隔组成大篇幅文本，所以我们可以简单的利用标点进行分句处理，利用空格进行分词处理。设计分句函数的思路很简单，英文文本中出现的标点一般为逗号“,”、句点“.”和问号“?”，假设我们有一段英文文本，可以先将文本按照句点分割成若干小段的文本，再将各小段文本按照内部出现的逗号或者问号再次切分。基于这样的思路，我们即使不利用外部的工具包，也能够通过调用 Python 中的内置函数来构建一个简单的英文分词器。

对于上结构比较简单的文本，可以自行编写函数进行处理，但是对于结构、内容更加复杂的英文文本，类似的操作就很难进行全面、细致的处理了。

作为基于 Python 的自然语言处理前沿平台，NLTK 为我们提供了一套更为专业的英文分词工具，相比于调用 Python 的内置函数，NLTK 的英文分词工具模式更加丰富，并且在去除停用词、词干化处理方面更为优秀

在 window 10 的 Anaconda Prompt 下安装



```
Anaconda Prompt
C:\Users\Administrator>pip install nltk
Collecting nltk
  Downloading nltk-3.2.5.tar.gz (1.2MB)
    100% |#####| 1.2MB 215kB/s
Requirement already satisfied: six in d:\program files\python\lib\site-packages (from nltk)
Installing collected packages: nltk
  Running setup.py install for nltk ... done
Successfully installed nltk-3.2.5
```

2.3.2 英文单词字符化

英文单词直接按照空格就可以分割开了，例如先处理第一个文档看看效果

```
In [13]: f = open('News_1_Org.txt', 'r').read()
        #f.readline()
        #for line in f:
        #    print line
        #A

In [14]: #把文本中每个单词字符化
        import re #正则表达式
        reg = re.compile(' \\W*') #除了单词外的所有特殊符号包括空格
        text = reg.split(f.read())
        print text[0:50]
        f.close()

['', 'Christopher', 'Manning', 'Thomas', 'M', 'Siebel', 'Professor', 'in', 'Machine', 'Learning', 'Professor',
'of', 'Linguistics', 'and', 'of', 'Computer', 'Science', 'Natural', 'Language', 'Processing', 'Group', 'Linguist
ics', 'and', 'Computer', 'Science', 'Stanford', 'University', 'Bio', 'Christopher', 'Manning', 'is', 'the', 'ina
ugral', 'Thomas', 'M', 'Siebel', 'Professor', 'in', 'Machine', 'Learning', 'in', 'the', 'Departments', 'of', 'Co
mputer', 'Science', 'and', 'Linguistics', 'at', 'Stanford']
```

单词全部转换成小写字母

```
In [17]: text_lower = [i.lower() for i in text] #单词全部转换成小写
print text_lower[0:100]

['', 'christopher', 'manning', 'thomas', 'm', 'siebel', 'professor', 'in', 'machine', 'learning', 'professor',
'of', 'linguistics', 'and', 'of', 'computer', 'science', 'natural', 'language', 'processing', 'group', 'linguist
ics', 'and', 'computer', 'science', 'stanford', 'university', 'bio', 'christopher', 'manning', 'is', 'the', 'ina
ugral', 'thomas', 'm', 'siebel', 'professor', 'in', 'machine', 'learning', 'in', 'the', 'departments', 'of', 'co
mputer', 'science', 'and', 'linguistics', 'at', 'stanford', 'university', 'his', 'research', 'goal', 'is', 'comp
uters', 'that', 'can', 'intelligently', 'process', 'understand', 'and', 'generate', 'human', 'language', 'materi
al', 'manning', 'is', 'a', 'leader', 'in', 'applying', 'deep', 'learning', 'to', 'natural', 'language', 'process
ing', 'with', 'well', 'known', 'research', 'on', 'tree', 'recursive', 'neural', 'networks', 'sentiment', 'analys
is', 'neural', 'network', 'dependency', 'parsing', 'the', 'glove', 'model', 'of', 'word', 'vectors', 'neural']
```

读取当前文本长度

```
In [4]: len(text_lower)

Out[4]: 1346
```

2.3.3 去除停用词

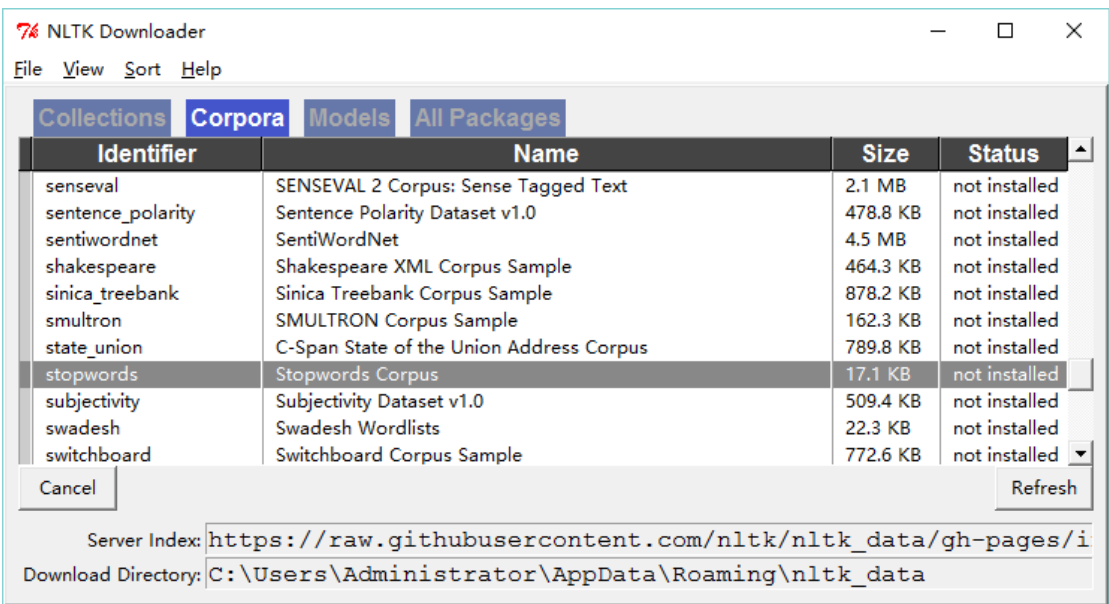
文本经过简单的而分词处理后，还会包含大量的无实际意义的通用词，需要过滤掉，NLTK 提供了一份英文停用词词典供使用者直接使用，可以通过以下方式查看停用词词典。

运行以下代码，打开下载工具下载 stopwords 停用词包

```
In [5]: import nltk
nltk.download()

showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

Out[5]: True
```



分词结果列表中还存在大量的标点符号和停用词,利用自定义标点符号列表以及 NLTK 提供的停用词典进行过滤,过滤后的分词结果保存在 `text_clear` 列表中。

导入该停用词表即可除去停用词,处理之后发现文本单词数从 1346 减少为 884

```
In [6]: from nltk.corpus import stopwords #nltk中的停用词表
en_stopwords = stopwords.words("english")
text_clear = []
for word in text_lower: #去掉停用词
    if word not in en_stopwords:
        text_clear.append(word)

print text_clear[0:50]

['', 'christopher', 'manning', 'thomas', 'siebel', 'professor', 'machine', 'learning', 'professor', 'linguistic
s', 'computer', 'science', 'natural', 'language', 'processing', 'group', 'linguistics', 'computer', 'science',
'stanford', 'university', 'bio', 'christopher', 'manning', 'inaugral', 'thomas', 'siebel', 'professor', 'machin
e', 'learning', 'departments', 'computer', 'science', 'linguistics', 'stanford', 'university', 'research', 'goa
l', 'computers', 'intelligently', 'process', 'understand', 'generate', 'human', 'language', 'material', 'mannin
g', 'leader', 'applying', 'deep']

In [7]: len(text_clear)

Out[7]: 884
```

2.3.4 抽取单词词干

利用 Porter 模块即 Porter Stemming 算法进一步进行词干化处理,将词干化后的结果保存在列表 `words_stem` 中。

因为 Porter Stemming 已经很经典地实现了,这里我搜索了网上开源的 python 算法,在这个网站里各种语言编写的算法都有:

<https://tartarus.org/martin/PorterStemmer/>

```
In [8]: #抽取词干
from nltk.stem.porter import PorterStemmer
st = PorterStemmer() #一个对象
text_stem = [st.stem(word) for word in text_clear]
for i in text_stem:
    print i
```

```
siebel
professor
machin
learn
professor
linguist
comput
scienc
natur
languag
process
group
linguist
```

最后将文件保存即可。用一个循环实现所有前 500 个文档的处理

```
In [12]: #保存
fp = open('News_1_E.txt', 'w')
for i in text_stem:
    fp.write(i + "\n")

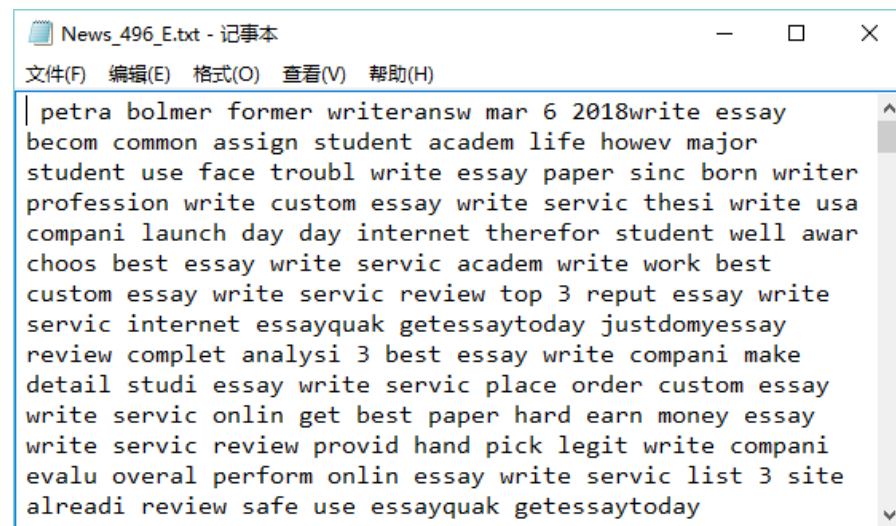
fp.close()

import re #正则表达式
num = 1
while(num <= 500):
    f = open('News_' + str(num) + '_Org.txt', 'r')

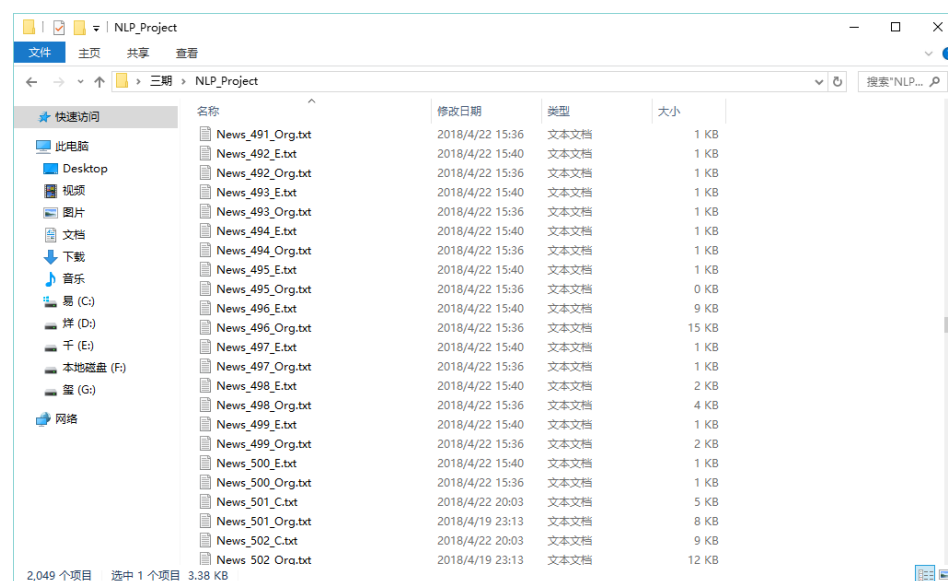
    #保存
    fp = open('News_' + str(num) + '_E.txt', 'w')
    for i in text_stem:
        fp.write(i + "\n")

    fp.close()
    num +=1
```

最后得到的清洗之后的文本如下：



petra bolmer former writeransw mar 6 2018write essay
becom common assign student academ life howev major
student use face troubl write essay paper sinc born writer
profession write custom essay write servic thesi write usa
compani launch day day internet therefor student well awar
choos best essay write servic academ write work best
custom essay write servic review top 3 reput essay write
servic internet essayquak getessaytoday justdomyessay
review complet analysi 3 best essay write compani make
detail studi essay write servic place order custom essay
write servic onlin get best paper hard earn money essay
write servic review provid hand pick legit write compani
evalu overall perform onlin essay write servic list 3 site
alreadi review safe use essayquak getessaytoday



名称	修改日期	类型	大小
News_491_Org.txt	2018/4/22 15:36	文本文档	1 KB
News_492_E.txt	2018/4/22 15:40	文本文档	1 KB
News_492_Org.txt	2018/4/22 15:36	文本文档	1 KB
News_493_E.txt	2018/4/22 15:40	文本文档	1 KB
News_493_Org.txt	2018/4/22 15:36	文本文档	1 KB
News_494_E.txt	2018/4/22 15:40	文本文档	1 KB
News_494_Org.txt	2018/4/22 15:36	文本文档	1 KB
News_495_E.txt	2018/4/22 15:40	文本文档	1 KB
News_495_Org.txt	2018/4/22 15:36	文本文档	0 KB
News_496_E.txt	2018/4/22 15:40	文本文档	9 KB
News_496_Org.txt	2018/4/22 15:36	文本文档	15 KB
News_497_E.txt	2018/4/22 15:40	文本文档	1 KB
News_497_Org.txt	2018/4/22 15:36	文本文档	1 KB
News_498_E.txt	2018/4/22 15:40	文本文档	2 KB
News_498_Org.txt	2018/4/22 15:36	文本文档	4 KB
News_499_E.txt	2018/4/22 15:40	文本文档	1 KB
News_499_Org.txt	2018/4/22 15:36	文本文档	2 KB
News_500_E.txt	2018/4/22 15:40	文本文档	1 KB
News_500_Org.txt	2018/4/22 15:36	文本文档	1 KB
News_501_C.txt	2018/4/22 20:03	文本文档	5 KB
News_501_Org.txt	2018/4/19 23:13	文本文档	8 KB
News_502_C.txt	2018/4/22 20:03	文本文档	9 KB
News_502_Org.txt	2018/4/19 23:13	文本文档	12 KB

2.4 中文文本处理

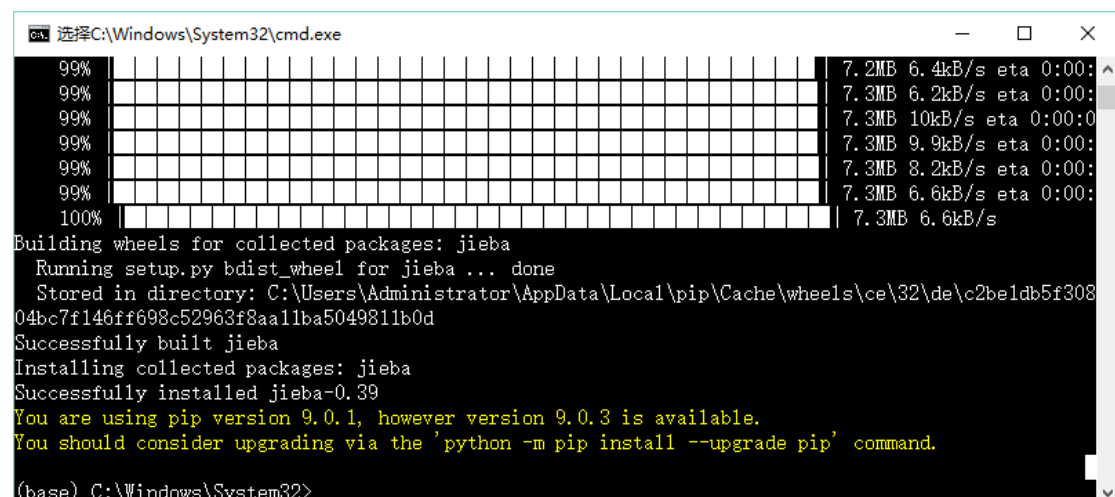
从形式上看，每一段中文文本都可以被认为是由汉字与标点符号组成的一个字符串。由字可组成词，由词可组成词组，由词组可组成句子，进而由一些句子组成段、节、章、篇。在我们的日常生活中，一个中文文本或是几个汉字的组合能够具有被赋予多种含义，作为人类，我们通常可以很容易地根据语境或是场景来对不同文本的语义进行理解，并从中分离出具有实际意义的词语，但对于计算机来说，能做到这一点则需要更多复杂的工作。Python 中常用的中文分词与词性标注工具，这些工具结合词库与算法在很大程度上解决了计算机的中文分词问题，掌握它们的使用方法，能给我们的文本数据分析带来极大的便利。

jieba 是 Python 中常用的中文分词与词性标注工具。对于给定的待分词文本，jieba 主要分词处理思路如下：

加载模块自带 dict.txt 词典，从自带的词典中构建待分词的语句的有向无环图（DAG）；对于词典中未收录的词，使用 HMM 模型的 Viterbi 算法尝试分词处理；已收录词和未收录词全部分词完毕后，根据有向无环图的最大概率路径使用 Python 的 yield 语法生成一个词语生成器，逐词语返回。

Jieba 的特点在于：支持多种分词模式——jieba 支持以下三种分词模式：精确模式，对语句进行最精确地切分，只用于文本数据分析；全模式，把句子中所有的可以成词的词语都扫描出来，该模式速度非常快，但是不能解决歧义；搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。支持繁体分词；支持自定义词典；待分词的字符串可以是 unicode 或 UTF-8 编码。

2.4.1 工具包安装



```
选择C:\Windows\System32\cmd.exe
99% | 7.2MB 6.4kB/s eta 0:00:
99% | 7.3MB 6.2kB/s eta 0:00:
99% | 7.3MB 10kB/s eta 0:00:0
99% | 7.3MB 9.9kB/s eta 0:00:
99% | 7.3MB 8.2kB/s eta 0:00:
99% | 7.3MB 6.6kB/s eta 0:00:
100% | 7.3MB 6.6kB/s
Building wheels for collected packages: jieba
  Running setup.py bdist_wheel for jieba ... done
  Stored in directory: C:\Users\Administrator\AppData\Local\pip\Cache\wheels\ce\32\de\c2be1db5f30804bc7f146ff698c52963f8aa11ba5049811b0d
Successfully built jieba
Installing collected packages: jieba
Successfully installed jieba-0.39
You are using pip version 9.0.1, however version 9.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
(base) C:\Windows\System32>
```

2.4.2 中文文本分词

```
In [97]: import jieba
        from jieba import posseg

        f = open('News_501_Org.txt')
        text = f.read()
        print text

注册一亩三分地论坛，查看更多干货！
您需要 登录 才可以下载或查看，没有帐号？ Sign Up 注册获取更多干货

x

写这个的主要目的，不是要阐述什么道理，争论什么话题，只是想以时间为序，把申请的大体过程和所要做的事情，串在一起，不求说法一定严谨，不求肯定适用每种情况，Warald只求给大家一个可以简单参考的大体步骤和计划表。步骤简述和大体时间：申请秋季入学（用本科生申请2018秋季入学为例）

出国其实从大一就应该开始准备，就算学校教育方式你很不喜欢、任课老师你觉得很肤浅，也得把功课学好，尤其是专业课
```

首先打开一个中文文档，调用 jieba 分词函数，用 ‘/’ 分开词语

```
In [98]: A=jieba.cut_for_search(text, HMM=False) #用jieba分词分割中英文
        B = "/".join(A) # 加入分隔符
        print B

/
/
/
/注册/一亩/三分/地/论坛/，/查看/更/多/干货/!/
/您/需要/ /登录/ /才/可以/下载/或/查看/，/没有/帐号/? /Sign/ /Up/ /注册/获取/更/多/干货/ /
/
/x/
/
/写/这个/的/主要/目的/，/不是/要/阐述/什么/道理/，/争论/什么/话题/，/只是/想/以/时间/为/序/，/把/申请/的/大体/
过程/和/所/要/做/的/事情/，/串/在/一起/，/不/求/说法/一定/严谨/，/不/求/肯定/适用/每种/情况/，/Warald/只求/给/
大家/一个/可以/简单/参考/的/大体/步骤/和/计划/计划表/。/步骤/简述/和/大体/时间/：/申请/秋季/入学/（/用/本科/本
科生/申请/2018/秋季/入学/为/例/）/
/
/出国/其实/从/大/一/就/应该/开始/准备/，/就算/学校/教育/方式/你/很/不/喜欢/、/任课/老师/任课老师/你/觉得/很/肤
浅/，/也/得/把/功/课/学/好/，/尤/其/是/专业/专业/课/（/选修/+ /必须/）/和/数学/数学/课/，/GPA/至少/80/（/这个/分数/
```

jieba.cut_for_search()函数是适用于搜索引擎构建倒排索引（Inverted index）的分词函数，调用方式为：jieba.cut_for_search(sentence, HMM=True)

sentence:需要分词处理的字符串；HMM:是否使用 HMM 模型，默认缺失值为 True，和 jieba.cut 一样，jieba.cut_for_search 返回的结果也是一个可迭代的 generator。

用正则表达式，以‘/’为分隔符把文本词语分到一个 List C，一个词语为 List C 中的一个元素

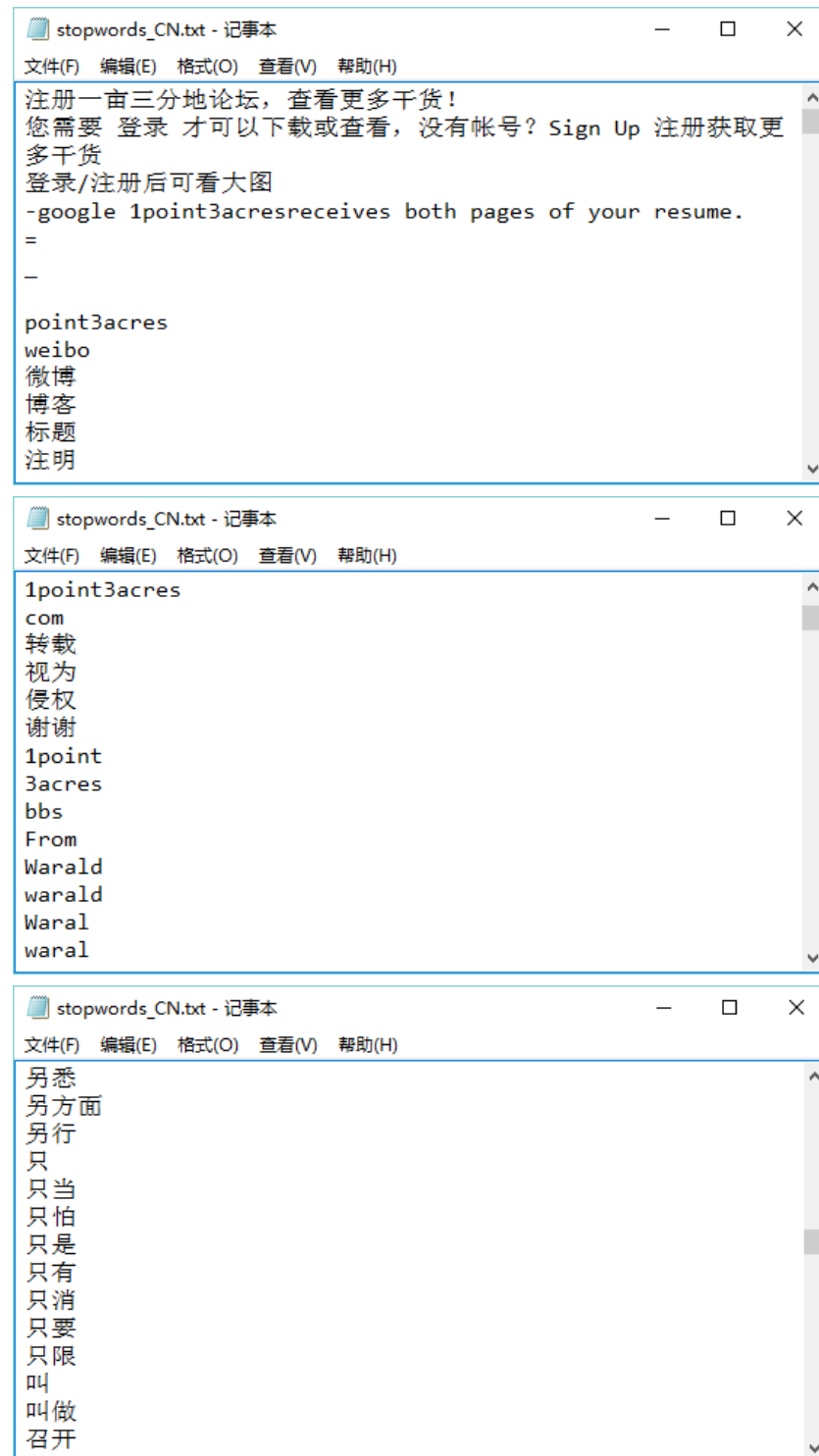
```
In [99]: import re
        reg = re.compile('/')
        C = reg.split(B)
        C #需要清洗的文本

Out[99]: [u' \n',
          u' \n',
          u' \n',
          u' \n',
          u' \u6ce8\u518c',
          u' \u4e00\u4ea9',
          u' \u4e09\u5206',
          u' \u5730',
          u' \u8bba\u575b',
          u' \uff0c',
          u' \u67e5\u770b',
          u' \u66f4',
          u' \u591a',
          u' \u5e72\u8d27',
```


2.4.3 构造中文停用词表

以上实现了中文文本的分词以及字符化，下面首先将停用词表同样分词和字符化，然后去除停用词。

因为我爬的是论坛内容，所以要去除一些论坛中的特有词组，比如 1poing3acres。自己构建的停用词表一部分为：



打开停用词表，进行同样的分词操作

```
In [100]: #以上实现了中文文本的分词以及字符化
#下面首先将停用词表同样分词和字符化
#然后去除停用词

In [101]: #jieba.cut_for_search 是适用于搜索引擎构建倒排索引 (Inverted index) 的分词函数,
#调用方式为: jieba.cut_for_search(sentence, HMM=True)
#返回的结果是一个可迭代的 generator
#jieba.lcut_for_search 可以直接返回列表结果
stopwordslist = open(' stopwords_CN.txt', 'r').read()

D = jieba.cut_for_search(stopwordslist, HMM=False) #用jieba分词分割中英文
E = "/".join(D) # 加入分隔符
print E

注册/ 一亩/三分/地/论坛/, /查看/更/多/干货/! /
/您/需要/ /登录/ /才/可以/下载/或/查看/, /没有/帐号/? /Sign/ /Up/ /注册/获取/更/多/干货/ /
/登录///注册/后/可/看大图/
/-/google/ /1point3acresreceives/ /both/ /pages/ /of/ /your/ /resume./
/=/
/_/
/_/
/point3acres/
```

用正则表达式，以‘/’为分隔符把文本词语分到一个 List F，一个词语为 List F 中的一个元素

```
In [102]: import re
reg = re.compile('/ ')

F = reg.split(E) #按照分隔符来划分单词为list
F #分割后的停用词表
u' \u5e72\u8d27',
u' \uff01',
u' \n',
u' \u60a8',
u' \u9700\u8981',
u' ',
u' \u767b\u5f55',
u' ',
u' \u624d',
u' \u53ef\u4ee5',
u' \u4e0b\u8f7d',
u' \u6216',
u' \u67e5\u770b',
u' \uff0c',
u' \u6ca1\u6709',
u' \u5e10\u53f7',
u' \uff1f',
```

2.4.4 去除中文停用词

```
In [103]: # 对句子去除停用词
def movestopwords(C):
    out = []
    for word in C:
        if word not in F:
            if word != '\t' and '\n':
                out.append(word)
    return out

clear = movestopwords(C)
print clear
#result = str(clear).replace('u|'', '|').decode("unicode-escape")
#print result
```

此时打印出的结果为 unicode 编码所以再进一步转换，使结果显示出中文字符

```
In [104]: print '/'.join(i for i in clear)
```

写/目的/阐述/道理/争论/话题/想/时间/序/申请/过程/做/事情/串/求/说法/严谨/求/肯定/每种/情况/只求/简单/参考/步骤/计划/计划表/步骤/简述/时间/申请/秋季/入学/本科/本科生/申请/2018/秋季/入学/例/出国/学校/教育/方式/喜欢/任课/老师/任课老师/肤浅/功课/学好/专业/专业课/选修/数学/数学课/GPA/至少/80/分数/低/建议/至少/考/85/排名/太/难看/有空/老师/做做/科研/发/论文/暑假/公司/做/实习/体会/喜欢/做/大三/大四/TG/英语/考试/英语考试/大四/学期/紧张/申请/阶段/申请/研究/研究生/英语/考试/英语考试/T/Toefl/托福/考上/100/分/奖学/奖学金/能力/有限/建议/使劲/使劲/使劲/低于/90/分/海/未必/免/托福/雅思/未必/托福/G/商学/学院/商学院/项目/GMAT/理工/工科/理工科/GRE/最新/新版/最新版/答案/GRE/TOEFL/IBT/托福/考试/托福考/托福考试/分/够用/分数/太/低/学校/申请/deadline/截止/日期/15/早/12/到来/月/申请/高/排名/学校/奖学/奖学金/11/月/解决/考试/情况/月/情况/蹭/到来/月/月/蹭/越/越/吃/蹭/蹭/发现/地上/一堆/offe
r/ad/报/建议/参考/申请/经验/经验谈/10G/11/月/考试/GRE/TOEFL/录取/过程/申请/专业/学位/申请/硕士/做好/全/自费/时间/读完/学位/费用/40/50/万/人民币/博士/绝大/绝大多数/全/奖/读完/时间/博士/就业/请/参考/PhD/正名/phd/博士/就业/硕士/硕士生/推荐/影响/硕士/就业/因素/硕士/就业/建议/订/地/主/跟踪/最新/美国/就业/信息/依赖/过时/脱离/脱离实
际/说法/定位/背景/申请/档次/学校/把握/较大/承受/能力/承受能力/容忍/风险/定位/只能/讲/概率/成败/概率/代表/成功/
概率/代表/失败/敬请/参考/定位/评估/分析/择校/系统/求/定位/请/班/规/里/统一/模板/提供/免费/定位/分析/服务/择校/
理智/选择/合适/最合适/综合/排名/综合排名/择校/事关/事关重大/DIY/仔细/研究/建议/择校/择校/tag/浏览/offer/ad/汇
报/选择/帖子/里/学校/信息/择校/工作/调/考/试/进度/放在/美国/学校/不甚了解/建议/至少/预留/星期/久/专心/研究/找/
学校/开出/官方/封口/盖章/成绩/成绩单/学校/排名/证明/在读/证明/optional/毕业/学位/证明/至少/翻译/一份/找/推荐/
推荐人/找/推荐/推荐人/推荐/推荐信/有用/一文/里/分析/几个/写作/PS/CV/REF/ps/样本/包括/好多/申请/名校/全/奖/duan
mupei/yi/领头/热心/免费/修改/简历/克利/伯克利/统计/博士/博士生/umich/ee/博士/博士生/zach/阵容/成/填写/网/申/交/
申请/申请费/一所/学校/50/150/美元/几种/网/申/系统/事项/注意事项/提供/介绍/网/申/填写/简单/比较简单/琐碎/有事/
问/TO/送/分/步/择校/建议/学校/网/申/电子/送/分/协议/速度/监督/推荐/推荐人/送/推荐/推荐信/邮寄/书面/材料/书面材

最后用一个循环的代码，把 501-100 的文档按照上述文本格式保存

```
import sys
reload(sys)
sys.setdefaultencoding("utf-8")
#保存
fp = open('News_' + str(num) + '_C.txt', 'w')
fp.write('/'.join(i for i in clear))

fp.close()
num +=1
```

News_501_C.txt - 记事本

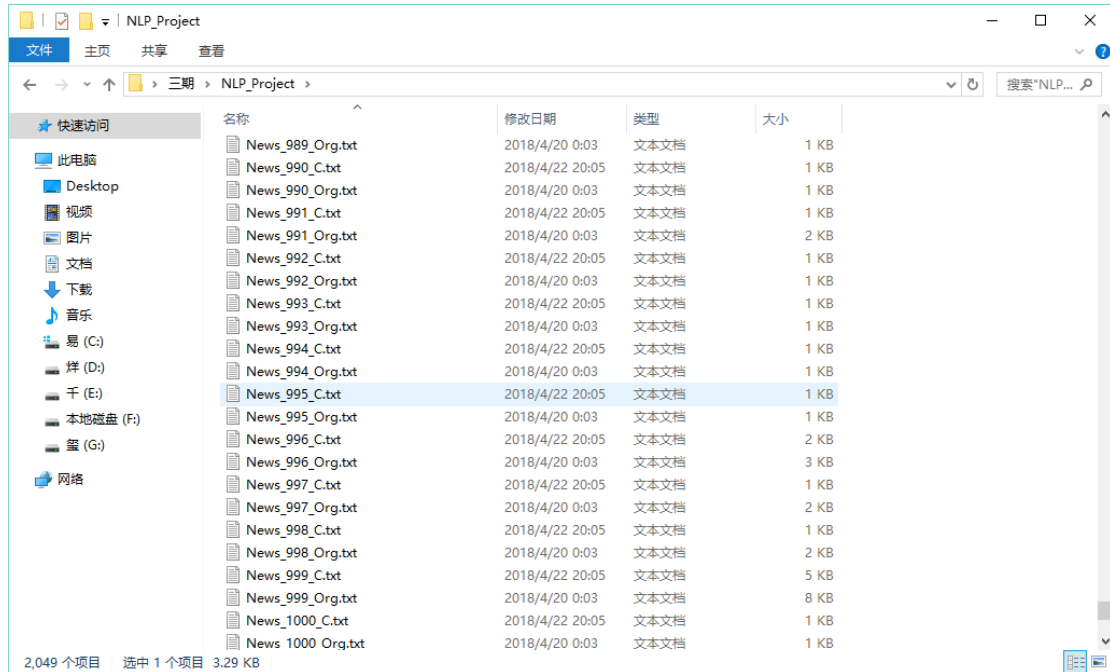
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

写/目的/阐述/道理/争论/话题/想/时间/序/申请/过程/做/事情/
串/求/说法/严谨/求/肯定/每种/情况/只求/简单/参考/步骤/计划/
计划表/步骤/简述/时间/申请/秋季/入学/本科/本科生/申
请/2018/秋季/入学/例/出国/学校/教育/方式/喜欢/任课/老师/任
课老师/肤浅/功课/学好/专业/专业课/选修/数学/数学课/GPA/至
少/80/分数/低/建议/至少/考/85/排名/太/难看/有空/老师/做做/
科研/发/论文/暑假/公司/做/实习/体会/喜欢/做/大三/大四/TG/
英语/考试/英语考试/大四/学期/紧张/申请/阶段/申请/研究/研
究生/英语/考试/英语考试/T/Toefl/托福/考上/100/分/奖学/奖学金
/能力/有限/建议/使劲/使劲/使劲/低于/90/分/海/未必/免/托福/
雅思/未必/托福/G/商学/学院/商学院/项目/GMAT/理工/工科/理工
科/GRE/最新/新版/最新版/答案/GRE/TOEFL/IBT/托福/考试/托福
考/托福考试/分/够用/分数/太/低/学校/申请/deadline/截止/日
期/15/早/12/到来/月/申请/高/排名/学校/奖学/奖学金/11/月/解

News_852_C.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

欧洲/申请/小结/选择/欧洲/理由/Master/申请/GRE/PhD/欧洲/一
份/工作/法律/最低/工资/低工资/最低工资/享受/福利/社会福利/
申请/压力/至少/Master/英国/荷兰/学校/学费/高/国家/学费/承
受/国家/申请/介绍/英国/英国/绝大/绝大多数/申请/申请者/接触
/美国/实际上/英国/内部/盟/校/出名/剑桥/牛津/帝国/master/奖
学/奖学金/英国/教育/产业/产业化/性价比/高/法国/实力/强/学
校/学校/规模/教学/体制/差异/名气/很大/法国/高/科/几个/公立
/大学/精英/学院/学/全免/费全免/法语/授课/导致/申请/难度/法
国/提供/国际/课程/工科/少/德国/申请/搜索/过程/涉及/德国/第
一/语言/不通/语言不通/德国/教育/体制/类似/本硕连/读/大
学/diploma/当于/相当于/硕士/德国/国际/硕士/课程/IMP/英语/
授课/学校/类似/课程/有人/拿到/ad/有没有/奖学/奖学金/太/学
费/北欧/四国/瑞典/挪威/丹麦/芬兰/北欧/四国/工/大学/学生/大



3. 参考网页

1. <https://tartarus.org/martin/PorterStemmer/>
2. <https://datartisan.gitbooks.io/begining-text-mining-with-python/content/>
3. <https://www.ranks.nl/stopwords>
4. <https://blog.csdn.net/u012052268/article/details/77825981>
5. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html#id37>
6. <https://blog.csdn.net/zzulp/article/details/77150129>
7. <http://qinxuye.me/article/porter-stemmer/>

4. 附录(整个工程源代码)

4.1

""""""英文网页 Quora""""""

```
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import urllib2
import warnings
warnings.filterwarnings("ignore")
```

```

web='https://www.quora.com/How-do-I-write-the-best-college-application-essay'
req = urllib2.Request(web)#request to visit web
page = urllib2.urlopen(req)#open this url
soup = BeautifulSoup(page, 'lxml')#将网页源码构造成 BeautifulSoup 对象，方便操作
table = soup.find_all("div",{"class" : "pagedlist_item"})#找到这个标签
URLquestion = []

for row in soup.find_all("li",{"class" : "related_question"}):
    for a in row.find_all('a', href=True):
        URLquestion.append(a['href'])

#加上完整的 url 链接
URLfull = []
for i in URLquestion:
    URLfull.append('https://www.quora.com' + i)

for url in URLfull:
    req = urllib2.Request(url)#request to visit web
    page = urllib2.urlopen(req)#open this url
    soup = BeautifulSoup(page, 'lxml')
    table2 = soup.find_all("div",{"class" : "pagedlist_item"})
    for t in table2:
        table.append(t)

import sys
reload(sys)
sys.setdefaultencoding("utf8")
num = 237
for content in table:
    if(len(content) == 0):
        continue;
    with open('News_' + str(num) + '_Org.txt', 'w') as f:
        f.write(content.get_text())
    num += 1
    if (num > 500):
        break;

""另一个英文网页 CSRanking""
f = open('URL-homepage.txt', 'r').read()
from bs4 import BeautifulSoup
import urllib2
import warnings

```

```
warnings.filterwarnings("ignore")
import sys
reload(sys)
sys.setdefaultencoding("utf8")

num = 227
with open('URL-homepage.txt', 'r') as f:#读取 url 链接
    for line in f:
        req = urllib2.Request(line)#request to visit web
        page = urllib2.urlopen(req)#open this url
        soup = BeautifulSoup(page, 'lxml')
        if(soup.body == None):
            continue;
        text = soup.body #body 里的文本
        with open('News_' + str(num) + '_Org.txt', 'w') as fp:
            fp.write(text.get_text())
            num += 1
```

4.2

```
""""""""""中文网页 一亩三分地""""""""""
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import urllib2
import warnings
warnings.filterwarnings("ignore")
#一亩三分地留学论坛精华帖
web='http://www.1point3acres.com/bbs/forum.php?mod=forumdisplay&fid=27&
filter=digest&digest=1'
req = urllib2.Request(web)#request to visit web
page = urllib2.urlopen(req)#open this url
soup = BeautifulSoup(page, 'lxml')
#所有页面的 url
Page = []
table = soup.find("div", "pg")#找到这个标签
for p in table.find_all('a', href = True):
    Page.append(p['href'])

import requests
URL = []
for i in soup.find_all('td', class_ = "icn"): #找帖子标题
    for a in i.find_all('a', href = True):
        URL.append(a['href'])
for num in range(0, 9):
    req = urllib2.Request(Page[num])#翻页
```

```

page = urllib2.urlopen(req)#open current url
soup = BeautifulSoup(page, 'lxml')
for i in soup.find_all('td', class_ = "icn"):
    for a in i.find_all('a', href = True):
        URL.append(a['href'])

import sys
reload(sys)
sys.setdefaultencoding( "utf8" )

import requests
num = 501
for url in URL:
    text = requests.get(url).text
    text = BeautifulSoup(text, from_encoding="gb18030")

    mainTags = text.find_all('td', {'class': "t_f"})
    if(len(mainTags) == 0):
        continue;
    with open('News_' + str(num) + '_Org.txt', 'w') as f:
        f.write(mainTags[0].get_text())
    num += 1

```

4.3

*****英文文本处理*****

```

import re #正则表达式
num = 1
while(num <= 500):
    f = open('News_' + str(num) + '_Org.txt', 'r')

    #把文本中每个单词字符化
    reg = re.compile('\W*')#除了单词外的所有特殊符号包括空格
    text = reg.split(f.read())
    #print text[0:50]
    f.close()

    text_lower = [i.lower() for i in text]#单词全部转换成小写
    #print text_lower[0:100]
    #len(text_lower)

    from nltk.corpus import stopwords #nltk 中的停用词表
    en_stopwords = stopwords.words("english")
    text_clear = []
    for word in text_lower: #去掉停用词

```

```

        if word not in en_stopwords:
            text_clear.append(word)

#print text_clear[0:50]
#len(text_clear)

#抽取词干
from nltk.stem.porter import PorterStemmer
st = PorterStemmer() #一个对象
text_stem = [st.stem(word) for word in text_clear]

#保存
fp = open('News_' + str(num) + '_E.txt', 'w')
for i in text_stem:
    fp.write(i + " ")

fp.close()
num +=1

```

4.4

*****中文文本处理*****

```

import jieba
from jieba import posseg
import re
num = 501
while(num <= 1000):
    f = open('News_' + str(num) + '_Org.txt', 'r')
    text = f.read()
    A=jieba.cut_for_search(text, HMM=False)#用 jieba 分词分割中英文
    B = "/" .join(A) # 加入分隔符

    reg = re.compile('/')
    C = reg.split(B)
    C #需要清洗的文本

    stopwordslist = open('stopwords_CN.txt', 'r').read()
    D = jieba.cut_for_search(stopwordslist, HMM=False)#用 jieba 分词分割中
英文
    E = "/" .join(D) # 加入分隔符

    F = reg.split(E) #按照分隔符来划分单词为 list
    F #分割后的停用词表

# 对句子去除停用词

```



```

def movestopwords(C):
    out = []
    for word in C:
        if word not in F:
            if word != '\t' and '\n':
                out.append(word)
    return out

clear = movestopwords(C)

import sys
reload(sys)
sys.setdefaultencoding( "utf-8" )
#保存
fp = open('News_' + str(num) + '_C.txt', 'w')
fp.write('/'.join(i for i in clear))

fp.close()
num +=1
4.5
"""
Porter Stemming 算法——来自官网 https://tartarus.org/martin/PorterStemmer/
"""

import sys

class PorterStemmer:

    def __init__(self):
        self.b = "" # buffer for word to be stemmed
        self.k = 0
        self.k0 = 0
        self.j = 0 # j is a general offset into the string

    def cons(self, i):
        if self.b[i] == 'a' or self.b[i] == 'e' or self.b[i] == 'i' or self.b[i]
== 'o' or self.b[i] == 'u':
            return 0
        if self.b[i] == 'y':
            if i == self.k0:
                return 1
            else:
                return (not self.cons(i - 1))
        return 1

```

```

def m(self):
    n = 0
    i = self.k0
    while 1:
        if i > self.j:
            return n
        if not self.cons(i):
            break
        i = i + 1
    i = i + 1
    while 1:
        while 1:
            if i > self.j:
                return n
            if self.cons(i):
                break
            i = i + 1
        i = i + 1
        n = n + 1
        while 1:
            if i > self.j:
                return n
            if not self.cons(i):
                break
            i = i + 1
        i = i + 1

def vowelinstem(self):
    for i in range(self.k0, self.j + 1):
        if not self.cons(i):
            return 1
    return 0

def doublec(self, j):
    if j < (self.k0 + 1):
        return 0
    if (self.b[j] != self.b[j-1]):
        return 0
    return self.cons(j)

def cvc(self, i):
    if i < (self.k0 + 2) or not self.cons(i) or self.cons(i-1) or not
self.cons(i-2):
        return 0

```

```

    ch = self.b[i]
    if ch == 'w' or ch == 'x' or ch == 'y':
        return 0
    return 1

def ends(self, s):
    length = len(s)
    if s[length - 1] != self.b[self.k]: # tiny speed-up
        return 0
    if length > (self.k - self.k0 + 1):
        return 0
    if self.b[self.k-length+1:self.k+1] != s:
        return 0
    self.j = self.k - length
    return 1

def setto(self, s):
    length = len(s)
    self.b = self.b[:self.j+1] + s + self.b[self.j+length+1:]
    self.k = self.j + length

def r(self, s):
    if self.m() > 0:
        self.setto(s)

def step1ab(self):
    """step1ab() gets rid of plurals and -ed or -ing. e.g.

```

```

    caresses -> caress
    ponies   -> poni
    ties     -> ti
    caress   -> caress
    cats     -> cat

```

```

    feed     -> feed
    agreed   -> agree
    disabled -> disable

```

```

    matting  -> mat
    mating   -> mate
    meeting  -> meet
    milling  -> mill
    messing  -> mess

```

```

        meetings -> meet
"""
if self.b[self.k] == 's':
    if self.ends("sses"):
        self.k = self.k - 2
    elif self.ends("ies"):
        self.setto("i")
    elif self.b[self.k - 1] != 's':
        self.k = self.k - 1
if self.ends("eed"):
    if self.m() > 0:
        self.k = self.k - 1
elif (self.ends("ed") or self.ends("ing")) and self.vowelinstem():
    self.k = self.j
    if self.ends("at"): self.setto("ate")
    elif self.ends("bl"): self.setto("ble")
    elif self.ends("iz"): self.setto("ize")
    elif self.doublec(self.k):
        self.k = self.k - 1
        ch = self.b[self.k]
        if ch == 'l' or ch == 's' or ch == 'z':
            self.k = self.k + 1
    elif (self.m() == 1 and self.cvc(self.k)):
        self.setto("e")

def step1c(self):
    if (self.ends("y") and self.vowelinstem()):
        self.b = self.b[:self.k] + 'i' + self.b[self.k+1:]

def step2(self):
    if self.b[self.k - 1] == 'a':
        if self.ends("ational"): self.r("ate")
        elif self.ends("tional"): self.r("tion")
    elif self.b[self.k - 1] == 'c':
        if self.ends("enci"): self.r("ence")
        elif self.ends("anci"): self.r("ance")
    elif self.b[self.k - 1] == 'e':
        if self.ends("izer"): self.r("ize")
    elif self.b[self.k - 1] == 'l':
        if self.ends("bli"): self.r("ble")
        elif self.ends("alli"): self.r("al")
        elif self.ends("entli"): self.r("ent")
        elif self.ends("eli"): self.r("e")
        elif self.ends("ousli"): self.r("ous")

```

```

elif self.b[self.k - 1] == 'o':
    if self.ends("ization"): self.r("ize")
    elif self.ends("ation"): self.r("ate")
    elif self.ends("ator"): self.r("ate")
elif self.b[self.k - 1] == 's':
    if self.ends("alism"): self.r("al")
    elif self.ends("iveness"): self.r("ive")
    elif self.ends("fulness"): self.r("ful")
    elif self.ends("ousness"): self.r("ous")
elif self.b[self.k - 1] == 't':
    if self.ends("aliti"): self.r("al")
    elif self.ends("iviti"): self.r("ive")
    elif self.ends("biliti"): self.r("ble")
elif self.b[self.k - 1] == 'g':
    if self.ends("logi"): self.r("log")

def step3(self):
    if self.b[self.k] == 'e':
        if self.ends("icate"): self.r("ic")
        elif self.ends("ative"): self.r("")
        elif self.ends("alize"): self.r("al")
    elif self.b[self.k] == 'i':
        if self.ends("iciti"): self.r("ic")
    elif self.b[self.k] == 'l':
        if self.ends("ical"): self.r("ic")
        elif self.ends("ful"): self.r("")
    elif self.b[self.k] == 's':
        if self.ends("ness"): self.r("")

def step4(self):
    if self.b[self.k - 1] == 'a':
        if self.ends("al"): pass
        else: return
    elif self.b[self.k - 1] == 'c':
        if self.ends("ance"): pass
        elif self.ends("ence"): pass
        else: return
    elif self.b[self.k - 1] == 'e':
        if self.ends("er"): pass
        else: return
    elif self.b[self.k - 1] == 'i':
        if self.ends("ic"): pass
        else: return
    elif self.b[self.k - 1] == 'l':

```

```

        if self.ends("able"): pass
        elif self.ends("ible"): pass
        else: return
    elif self.b[self.k - 1] == 'n':
        if self.ends("ant"): pass
        elif self.ends("ement"): pass
        elif self.ends("ment"): pass
        elif self.ends("ent"): pass
        else: return
    elif self.b[self.k - 1] == 'o':
        if self.ends("ion") and (self.b[self.j] == 's' or self.b[self.j]
== 't'): pass
        elif self.ends("ou"): pass
        else: return
    elif self.b[self.k - 1] == 's':
        if self.ends("ism"): pass
        else: return
    elif self.b[self.k - 1] == 't':
        if self.ends("ate"): pass
        elif self.ends("iti"): pass
        else: return
    elif self.b[self.k - 1] == 'u':
        if self.ends("ous"): pass
        else: return
    elif self.b[self.k - 1] == 'v':
        if self.ends("ive"): pass
        else: return
    elif self.b[self.k - 1] == 'z':
        if self.ends("ize"): pass
        else: return
    else:
        return
    if self.m() > 1:
        self.k = self.j

def step5(self):
    self.j = self.k
    if self.b[self.k] == 'e':
        a = self.m()
        if a > 1 or (a == 1 and not self.cvc(self.k-1)):
            self.k = self.k - 1
    if self.b[self.k] == 'l' and self.doublec(self.k) and self.m() > 1:
        self.k = self.k - 1

```

```

def stem(self, p, i, j):

    self.b = p
    self.k = j
    self.k0 = i
    if self.k <= self.k0 + 1:
        return self.b
    self.step1ab()
    self.step1c()
    self.step2()
    self.step3()
    self.step4()
    self.step5()
    return self.b[self.k0:self.k+1]


if __name__ == '__main__':
    p = PorterStemmer()
    if len(sys.argv) > 1:
        for f in sys.argv[1:]:
            infile = open(f, 'r')
            while 1:
                output = ''
                word = ''
                line = infile.readline()
                if line == '':
                    break
                for c in line:
                    if c.isalpha():
                        word += c.lower()
                    else:
                        if word:
                            output += p.stem(word, 0, len(word)-1)
                            word = ''
                        output += c.lower()
                print output,
            infile.close()

```