中国科学院大学
University of Chinese Academy of Sciences

自 然 语 言 处 理

技术报告

班级：2023 春季自然语言处理班

组号：--

姓名：唐嘉良

学号：2020K8009907032

报告主题：计算中英文信息源的熵

2023 年 4 月 2 日

一. 报告摘要

本次任务中，我采用 python 语言，利用 BeautifulSoup 等库进行爬虫、数据清洗和熵计算。

针对中文熵计算，选取了 https://news.163.com/（网易新闻），https://baike.baidu.com/（百度百科），http://www.xbiquge.la/10/10489/（笔趣阁）三个主网站；

针对英文熵计算，选取了 https://www.npr.org/，https://english.cctv.com/，https://www.audible.com/三个主网站。

随后利用双层 DFS 搜索算法来检索次级网站并爬取数据，采用正则表达式匹配方法进行数据清洗。最后通过汉字统计计算中文熵、通过英文字母和空格统计计算英文熵。

二. 网络爬虫与浅层 DFS 爬虫算法

采用 requests，BeautifulSoup 等库进行爬虫，用 request.get 函数获取网页内容，指定编码并利用 BeautifulSoup 解析格式。具体代码如下：

```
u = requests.get(r'https://www.npr.org/')
u.encoding = 'utf-8'
web = u.text
soup = BeautifulSoup(web, "lxml")
```

不同的网页有不同的特征，网址主页往往包括海量的次级链接。基于这一观察，采用如下 2 层 DFS 算法搜索主网页存放的次级网页网址信息。

首先搜索主网页所有子网页信息：

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

all_url=[] #存放第一层爬虫网址
all_double_url=[] #所有爬过的网址
# header = {"User-Agent":r"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM
u = requests.get(r'https://baike.baidu.com/')
u.encoding = 'utf-8'
web = u.text
soup = BeautifulSoup(web, "lxml")
for link in soup.find_all(name='a', href=re.compile(r'https?://baike.baidu.com/+')):
    if len(all_url) > max_num:
        break
    url = link.get('href')
    if url in all_url:
        continue
    all_url.append(url)
    all_double_url.append(url)
    print(len(all_url))
    print(url)
    #req = requests.get(url=url,headers = header)
    req = requests.get(url=url)
    req.encoding = 'utf-8'
    html = req.text
    bes = BeautifulSoup(html,"lxml")
    texts_list = bes.text.split("\xa0"*4)
    for line in texts_list:
        file.write(line+"\n")
```

将子网页作为主网页，依次再度搜寻其子网页信息：

```
for uurl in all_url:
    double_url=[]
    u = requests.get(url=uurl)
    u.encoding = 'utf-8'
    web = u.text
    soup = BeautifulSoup(web, "lxml")
    for link in soup.find_all(name='a', href=re.compile(r'https?://baike.baidu.com/+')):
        if len(double_url) > max_num:
            break
        url = link.get('href')
        if url in all_double_url :
            continue
        double_url.append(url)
        all_double_url.append(url)
        print(len(double_url))
        print(url)
        #req = requests.get(url=url,headers = header)
        req = requests.get(url=url)
        req.encoding = 'utf-8'
        html = req.text
        bes = BeautifulSoup(html,"lxml")
        # texts = bes.find("div", id = "container") #注意id是否是想要的
        # texts_list = texts.text.split("\xa0"*4)
        texts_list = bes.text.split("\xa0"*4)
        # file = open("D:\chtxt.txt","w",encoding='utf-8')   ##打开读写文件，逐行将列表读入文件内
        for line in texts_list:
            file.write(line+"\n")
```

搜寻过程中以 max_num 控制每一层搜寻广度上界。调整 max_num 可以以更高的时间开销爬虫更多文本。此外，搜寻过程中需要进行网址去重，确保网页信息不会重复爬取。

某些特殊网页的特征明显。例如，针对小说网站目录页类型的主页，其次级网页一般不会有太多信息质量较高的次级网页，只需进行一层搜索，确保有效信息密度，减少时间开销。

*注：“浅层 DFS 算法”是作者自己起的名字

三．数据清洗

将所有爬虫数据保存至 txt 文件中，随后进行数据清洗。

针对中文文本，利用正则表达式，结合汉字的 GTF-8 编码范围，去除除了汉字之外的所有字符，并写回文本文件：

```
# 清洗爬虫下来的数据，计算熵并打印输出
# 打开文件
fr=open('D:\chtxt.txt','r', encoding='utf-8')
# 去掉英文字母、标点符号，只留下汉字
txt = fr.read()
# 使用正则表达式去匹配标点符号
# txt = re.sub("[0-9a-zA-Z\s+\.\!\/_,$%^*(+\"\')]+|[+——()?【】“”！，。？、~@#￥%……&* （）：《》「」|:·+-]", "",txt)
# txt = txt.replace(" ","")
# txt = txt.replace("\n","")
# txt = txt.replace("\t","")
# txt = txt.replace("\r","")
txt = re.sub(r'[^\u4e00-\u9fa5]+', "",txt)
fr.close()

#把清洗过的文本写回去
w = open('D:\chtxt.txt','w', encoding='utf-8')
w.write(str(txt))
w.close()
```

针对英文文本，利用正则表达式，去除除了英文字母以及空格之外的所有字符，将小写字母全部转化为大写字母以便统计，并写回文本文件：

```python
# 打开文件
fr=open('D:\entxt.txt','r', encoding='utf-8')
# 去掉汉语、特殊符号，只留下英文字母
txt = fr.read()
# 使用正则表达式去匹配特殊符号
# txt = re.sub("[0-9\.\!\/_,\$%\^\*\(\+\"\'\)]|[—\?【】“”！，。？、~@#￥%……&\* （）：《》「」\|:\·\-\=<>‘’~@×O\[\]]", "",txt)
txt = re.sub('[^A-Za-z|\s]+','',txt)
# pattern = re.compile(r'[\u4e00-\u9fa5]+')
# txt = re.sub(pattern,"",txt) #排除汉字
txt = re.sub('[\n|\t|\r|\f|\v]+','',txt)
# txt = txt.replace("\n","")
# txt = txt.replace("\t","")
# txt = txt.replace("\r","")
txt = txt.upper()
# txt = txt.replace("[^A-Za-z ]+","")
fr.close()

#把清洗过的文本写回去
w = open('D:\entxt.txt','w', encoding='utf-8')
w.write(str(txt))
w.close()
```

四．字母/汉字出现特征统计及熵的计算

针对中文，我们首先选取 max_num=10，清洗后得到 428KB 语料，统计各个汉字出现频率，计算得到熵约为 9.86bit：

```
语料库共有145887个字
其中有2870个不同的字
Entropy of Chinese is 9.861126
```

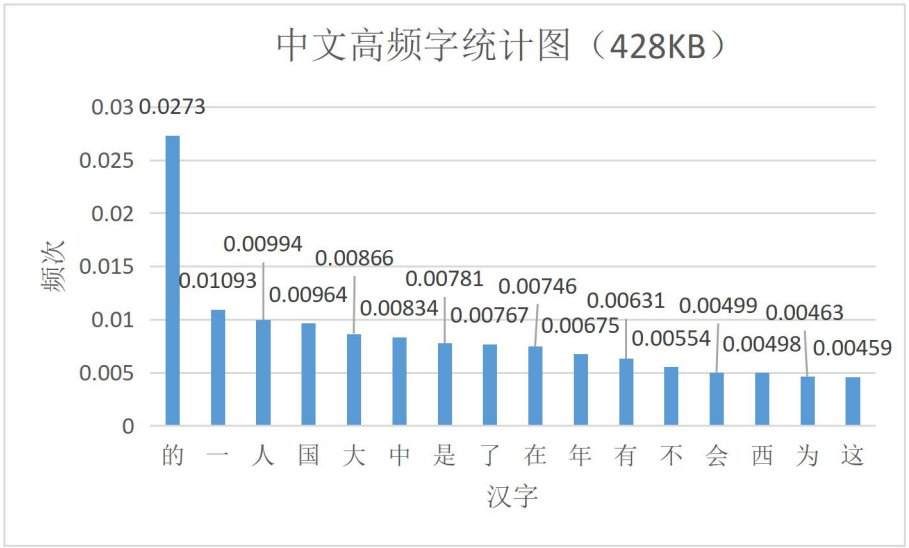
针对英文，我们首先选取 max_num=10，清洗后得到 2110KB 语料，统计各个汉字出现频率，计算得到熵约为 3.95bit：

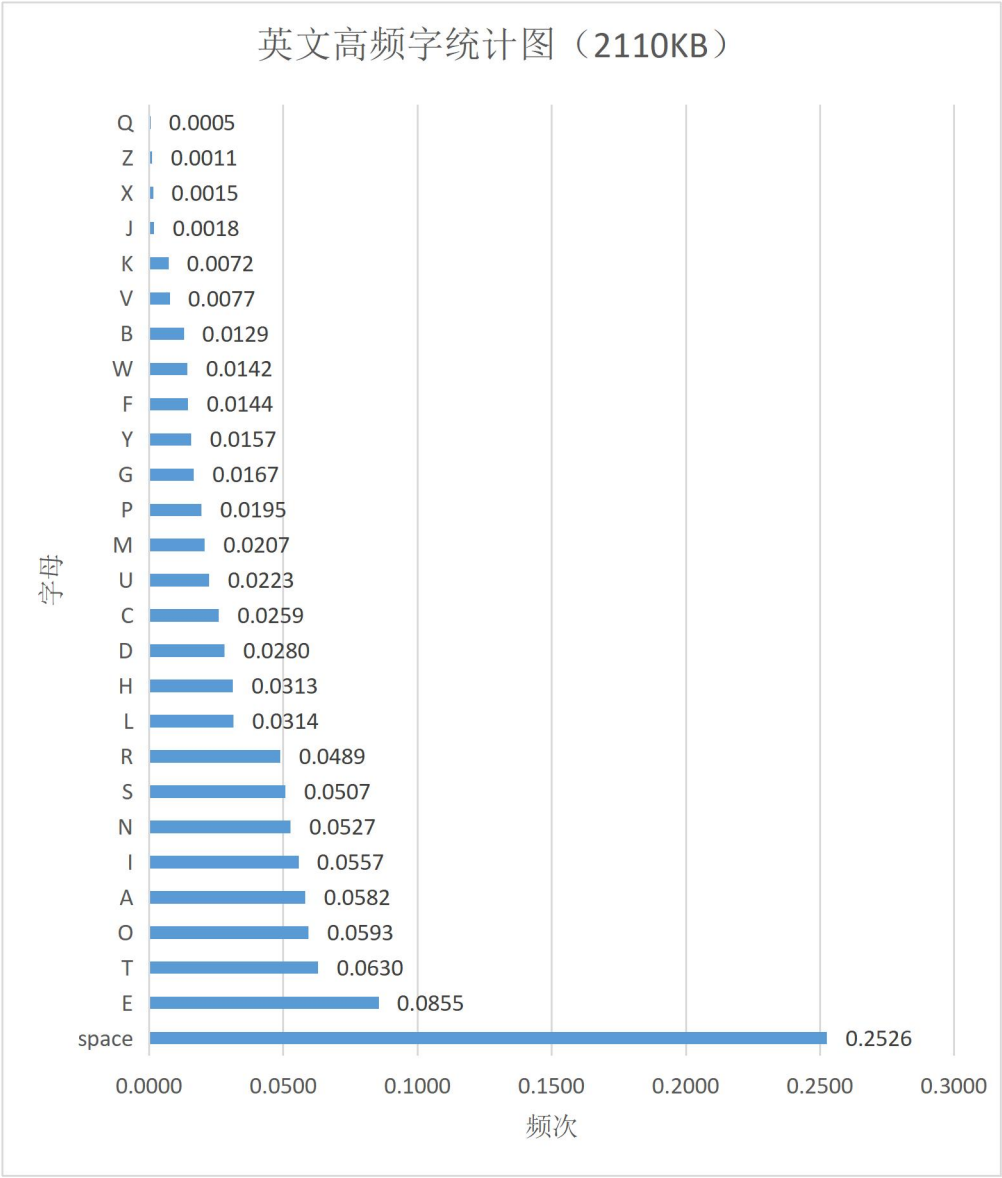
```
语料库共有1772416个字母
其中有28个不同的字母
```

```
Entropy of English is 3.946826
```

*注：英文文本字母总数不是 27 的原因是有特殊空字符的出现，这使得正则表达式无法排除它们。但它们出现频次极低，不影响计算结果。

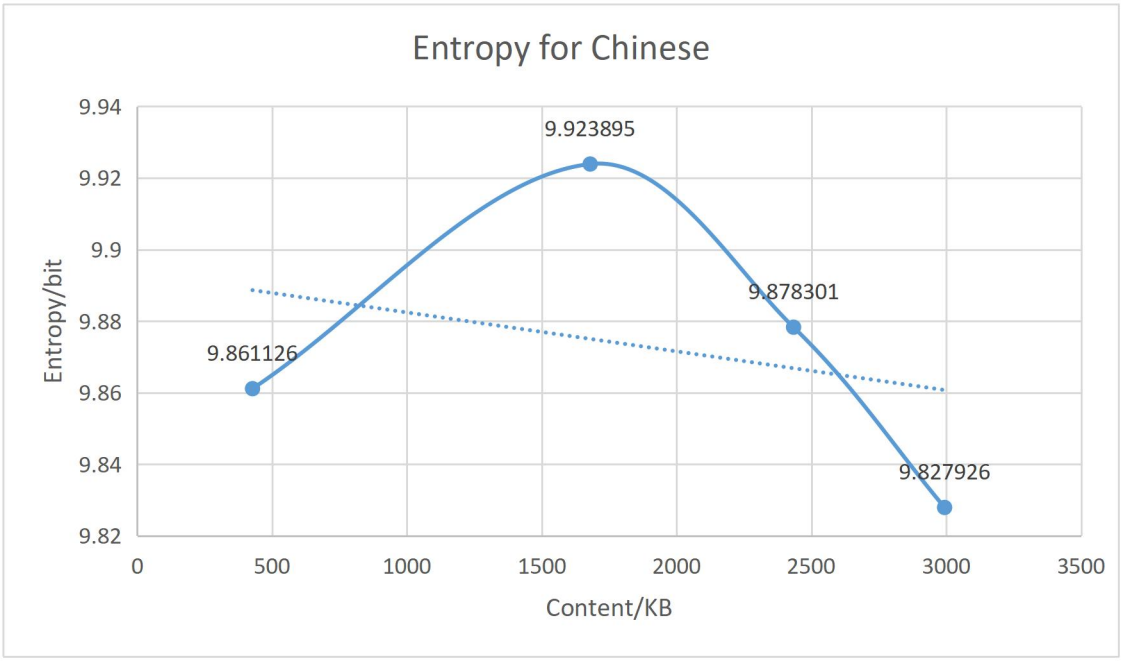
计算高频汉字/字母出现频次如下：

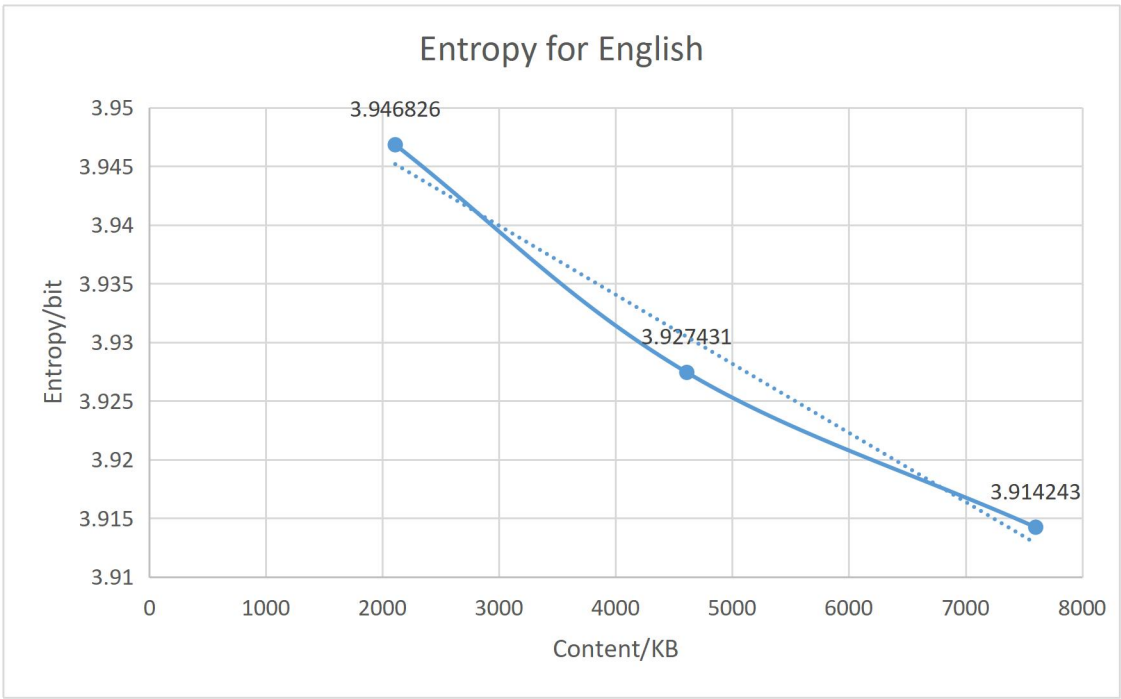




下面我们扩大语料库并计算熵，观察熵的变化。针对语料库文本文件自身大小的精确调整是困难的，因为不同网页的信息含量各不相同，我们所能做的只是精确地改变 max_num，从搜索广度的角度完成语料库的逐步扩大。

针对中文，改变 max_num 分别为 50，100，200；针对英文，改变 max_num 分别为 20，30。统计熵的变化随语料库大小变化及趋势如下所示：



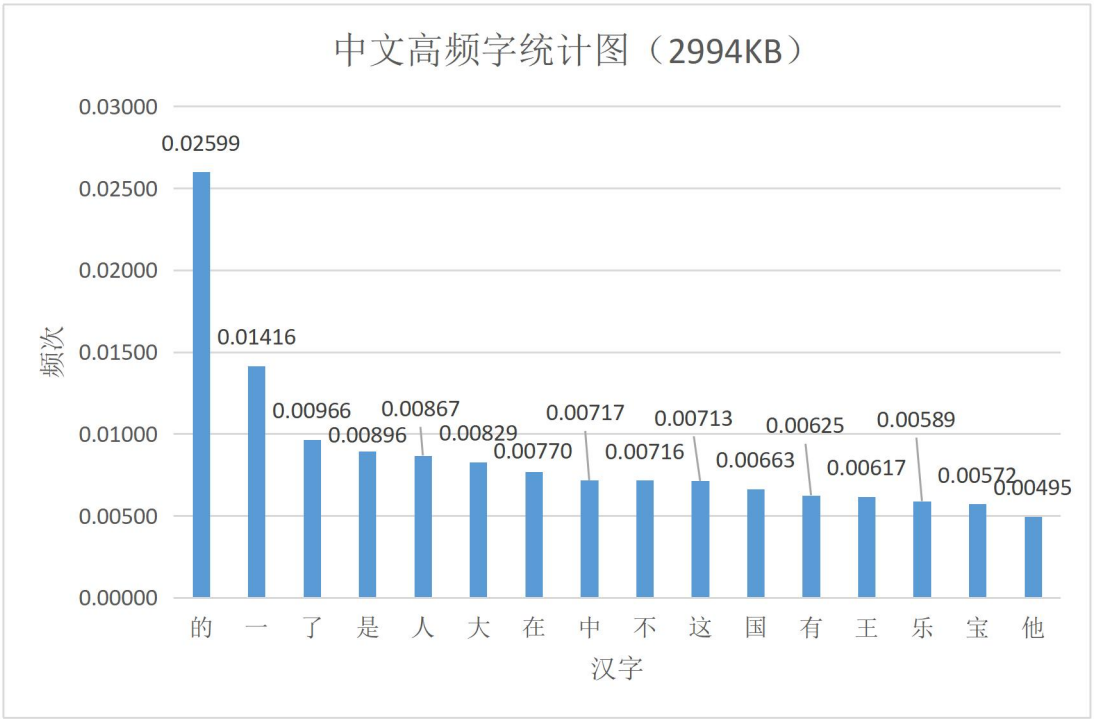


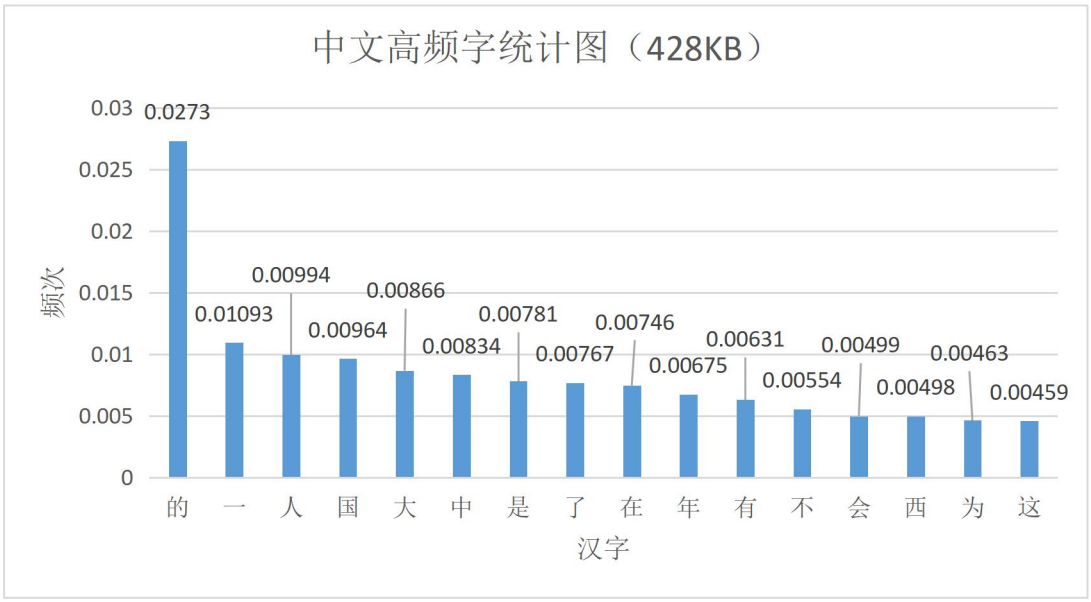
可以看出，扩大语料库之后中英文的熵值有所波动，逐渐向理论值靠近。在最大的语料库下，中英文熵值分别约为 9.83bit 和 3.91bit。

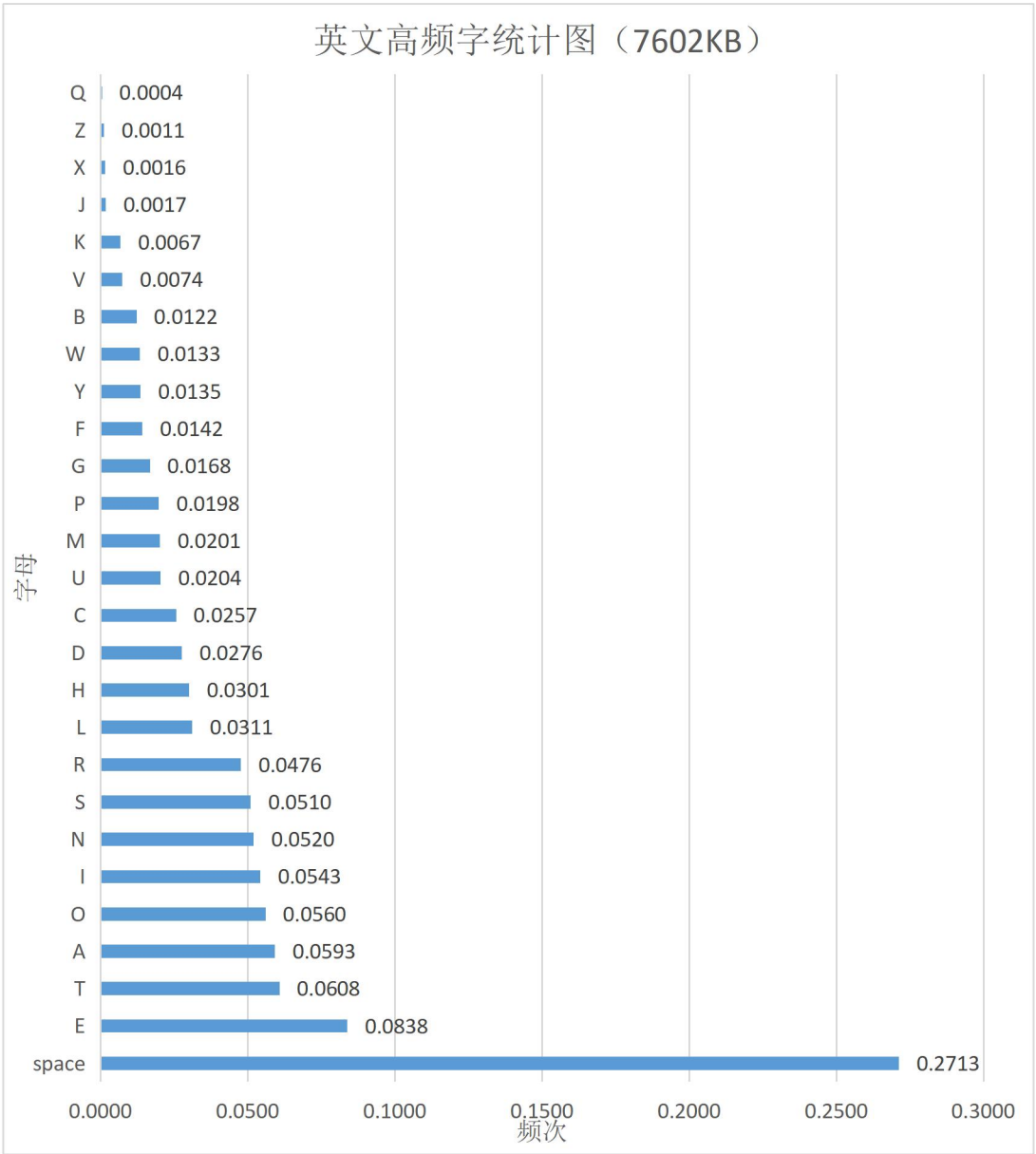
分别对 2994KB 的中文语料库和 7602KB 的英文语料库进行字母/汉字频率统计，并与 428KB 的中文语料库和 2110KB 的英文语料库对比：

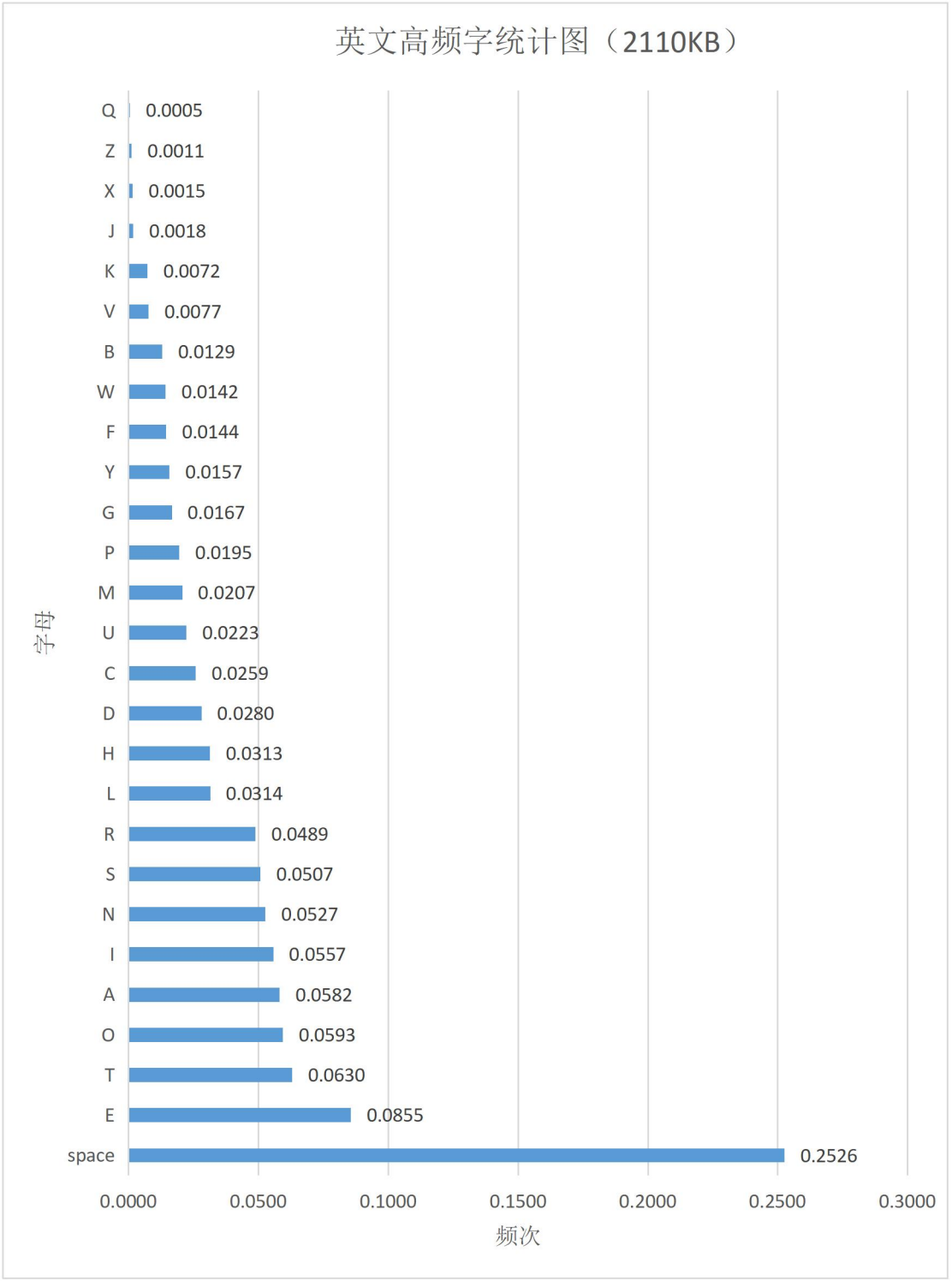
```
语料库共有1038402个字
其中有4319个不同的字
[ 的 ] 共出现   26989 次
[ 一 ] 共出现   14703 次
[ 了 ] 共出现   10032 次
[ 是 ] 共出现   9310 次
[ 人 ] 共出现   9001 次
[ 大 ] 共出现   8606 次
[ 在 ] 共出现   7998 次
[ 中 ] 共出现   7444 次
[ 不 ] 共出现   7438 次
[ 这 ] 共出现   7399 次
[ 国 ] 共出现   6886 次
[ 有 ] 共出现   6490 次
[ 王 ] 共出现   6404 次
[ 乐 ] 共出现   6113 次
[ 宝 ] 共出现   5935 次
[ 他 ] 共出现   5144 次
```

```
语料库共有6405471个字母
其中有30个不同的字母
[   ] 共出现   1737885 次
[ E ] 共出现   536654 次
[ T ] 共出现   389148 次
[ A ] 共出现   379775 次
[ O ] 共出现   358781 次
[ I ] 共出现   347930 次
[ N ] 共出现   333267 次
[ S ] 共出现   326629 次
[ R ] 共出现   305022 次
[ L ] 共出现   199341 次
[ H ] 共出现   193023 次
[ D ] 共出现   176869 次
[ C ] 共出现   164446 次
[ U ] 共出现   130565 次
[ M ] 共出现   128792 次
[ P ] 共出现   126766 次
[ G ] 共出现   107546 次
[ F ] 共出现   90639 次
[ Y ] 共出现   86606 次
[ W ] 共出现   85132 次
[ B ] 共出现   78299 次
[ V ] 共出现   47354 次
[ K ] 共出现   43160 次
[ J ] 共出现   10985 次
[ X ] 共出现   10299 次
[ Z ] 共出现   7142 次
[ Q ] 共出现   2848 次
```




中文高频字统计图（428KB）
0.03
0.025
0.02
0.015
0.01
0.005
0
频次
0.0273
0.01093
0.00994
0.00964
0.00866
0.00834
0.00781
0.00767
0.00746
0.00675
0.00631
0.00554
0.00499
0.00498
0.00463
0.00459
的 一 人 国 大 中 是 了 在 年 有 不 会 西 为 这
汉字


英文高频字统计图（7602KB）
字母
Q 0.0004
Z 0.0011
X 0.0016
J 0.0017
K 0.0067
V 0.0074
B 0.0122
W 0.0133
Y 0.0135
F 0.0142
G 0.0168
P 0.0198
M 0.0201
U 0.0204
C 0.0257
D 0.0276
H 0.0301
L 0.0311
R 0.0476
S 0.0510
N 0.0520
I 0.0543
O 0.0560
A 0.0593
T 0.0608
E 0.0838
space 0.2713
0.0000 0.0500 0.1000 0.1500 0.2000 0.2500 0.3000
频次



可以看到，中英文扩充语料库前后的汉字/字母频次没有发生根本性的改变，整体格局保持一致，并且其频次数据很符合经验与直觉。但是英文空格有点太多了，这与 PPT 上结论不一致，推测是因为网站信息有许多自带冗余空格；此外，汉字“宝”出现频次过高，这应该是因为所爬虫的小说网站中主角名字里含有“宝”字。除此以外所得数据结论符合预期，计算得到的中英文熵与 PPT 结论相比差别很小，说明本研究数据具备一定正确性和代表性。

参考资料

[1]https://blog.csdn.net/weixin_39525526/article/details/120502728
[2]https://blog.csdn.net/weixin_54852327/article/details/115916146?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-115916146-blog-118634078.235^v27^pc_relevant_3mothn_strategy_recovery&spm=1001.2101.3001.4242.1&utm_relevant_index=3
[3]https://cloud.tencent.com/developer/article/1478184