汉字与字母熵技术报告

谷朝阳 2020K8009929038

一、 研究目的

本研究旨在通过爬虫等资源获取工具，得到足够的字母和汉字数据，探究英文字母和中文汉字在文本中出现的频率以及各自熵的大小，并研究在不同文本规模下二者熵的差异，分析其产生的原因。

二、 研究方法

1. 数据来源

为了对比不同数据样本之间熵的差异，中文样本在全本小说网和人民网上获取，英文样本在 20news 数据集和 aclImdb 数据集中获取。样本覆盖了日常白话用语（小说和影评）以及正式用语（新闻）。

2. 数据采集

（1） 汉字

首先是汉语的数据采集，我使用 python 爬虫了全本小说网和人民网中的汉字资源。最终两个网站各自获得了超过一千五百万的汉字数据。

下对爬虫代码作简要介绍：

```
url = "https://www.xqb5200.com/87_87285/"
header = {"User-Agent":
            "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/70.0.3538.25 "
            "Safari/537.36 Core/1.70.3861.400 QQBrowser/10.7.4313.400"}
req = requests.get(url=url, headers=header)
req.encoding = "gbk"
```

url 是我们要爬虫的网站地址；这里的 header 是我们用于隐藏自己的一些配置，是有固定形式的，后使用 requests 库传入参数即可；最后根据网站内容设置编码格式。

```
html = req.text
bes = BeautifulSoup(html, "lxml")
texts = bes.find("div", id="list")
chapters = texts.find_all("a")
words = []
print(chapters)
```

然后使用 BeautifulSoup 库，在相应的网页上查找具体要爬的信息。这里就需要在网页上右击打开检查功能，从网站的结构里寻找正文的模块，将其 id 或者 class 传入 find 函数，才能找到对应文章。

我总共编写了六个爬虫程序，其中第一个是用于爬小说网站的，后面五个都是爬人民网的文章。人民网中资源较为分散，汉字资源很难聚集，所以需要在不同网页上寻找，而每次寻找的量都很少，最后我在“习近平系列重要讲话数据库”中获得了较多聚集的文章内容，丰富了最后的文本数据资源。

在 func6.py 中，利用 requests 还需要多传入一个 params，是因为其中翻页是通过网页代码内嵌实现的，而没有改变网址，故需要在网络请求的时候模仿修改其中的 page，进而实现等同于翻页的功能。

```
params = {
    'page': page
}
req = requests.get(url=url, params=params, headers=header)
```

（2）字母

英文文章由于代理网络等问题，难以直接采用爬虫实现。我曾经使用过一些英文数据集，这些数据集具有普适性，且内容充足，能成为最终的英文数据库。

我使用了 20news 数据集和 aclImdb 数据集中的内容，前者是大量新闻的数据集，后者是评论情感分析的数据集，前者文字较为官方正式，后者文字较为白话，具有区分性。

3. 数据处理

在汉语数据的采集过程中，首先要注意一些编码问题，例如’\xa0’等编码在网页上是呈现的空格，但是无法写入对应文件，故需要利用 replace 函数直接转化成空：

```
line = line.replace(" ", "").replace("\n", "")
```

除此之外，网站上还是有一些奇怪的编码，在爬虫过程中很容易卡死报错，所以在遇见编码问题的时候，可以使用 try except，跳过存在编码问题的文件，继续下一个文件的读取编写。

在小说 txt 中，数据查看时发现第一行都是推广该网站的广告信息，这会影响最终的文字熵，故通过代码删去：

```
load_path = r'F:\novels\Dou2'
files = glob.glob(load_path + '\\*.txt')
for file in tqdm(files):
    with open(file, 'r', encoding='gbk') as f:
        lines = f.readlines()

    # 删除第一行
    lines.pop(0)

    # 将处理后的内容写回到原文件中
    with open(file, 'w', encoding='utf-8') as f:
        f.writelines(lines)
```

```
100%|██████████| 1856/1856 [00:01<00:00, 1680.68it/s]
```

这里顺便将 gbk 编码转化为 utf-8，写回原文件。

在处理小规模数据的时候，列举出现概率最高的 20 个汉字的时候，发现了一些意外结果，即小说中主角的名字，所以简单设计了一个消除小说中主角人名的函数：

```
# 删除人名
import re
load_path = r'F:\novels\Dou'
re_name = 'XXX'
files = glob.glob(load_path + '\\*.txt')
for file in tqdm(files):
    with open(file, 'r', encoding='utf-8') as f:
        text = f.read()
    text = re.sub(re_name, '', text)
    with open(file, 'w', encoding='utf-8') as f:
        f.write(text)
```

但是随着数据库的规模变大，这项操作也变得无关紧要，因为不同小说的主角不同，最终综合下来也很难影响文字熵。

在英文数据集中，发现了大量的< br/>，这是表示回车的符号，可以直接删去，防止 b 和 r 两个字母对整体的影响。

```
# 定义函数，用于删除字符串中的 <br> 标签
def remove_br_tags(text):
    return text.replace("<br />", "")
```

处理英文数据的时候，要注意其中包含一些大写和小写字母，这里需要将其归一化，记作相同的字母即可，统一用一种形式表示。

在处理 20news 数据集时，出现了编码问题，读取文件的时候一直报错 0xff 不在 utf-8 编码内，这里我怀疑是数据集中包含了不同的编码文件，故我使用 try except 跳过了部分报错，结果表明绝大多数数据文件都是 utf-8 编码的。

代码中采用直接判断字符是否为汉字或者字母的方法，故标点符号和数字等不计入统计。

自此，中文和英文的数据集都搭建完成。

4. 公式计算

对于概率的计算，我采用了字典和 collections 库里的 Counter 进行统计处

理。只需要统计每个字或者字母出现次数，最后根据字典的排序即可排列出出现概率最多的汉字或者字母。

汉字中枚举出现概率最高的 20 个字，英文字母中选择出现最多的 5 个字母。

对于熵的计算，主要取决于上述出现的概率，公式如下：

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(xi)$$

5. 递增策略

汉字采取的递归策略是从一百万字开始，每次递增一百万个汉字，直至一千五百万字，统计概率和熵。

字母采取的递归策略是从一百万个单词开始，每次递增一百万个单词，直至两千万个，统计概率和熵。

三、 研究结果

1. 汉字

汉字结果如下所示：

	1000000	2000000	3000000	4000000	5000000	6000000	7000000	8000000	9000000	1E+07	1.1E+07	1.2E+07	1.3E+07	1.4E+07	1.5E+07
概率字	的	的	的	的	的	的	的	的	的	的	的	的	的	的	的
自上而下	国	国	国	国	国	国	国	国	国	国	国	国	国	国	国
概率减小	中	中	中	中	中	中	中	中	中	中	中	中	中	中	中
	和	和	和	和	和	和	和	和	和	和	和	和	和	和	和
	人	人	人	人	人	人	人	人	人	人	人	人	人	人	人
	要	要	一	要	发	发	发	发	发	发	要	要	要	要	要
	一	一	要	一	要	要	要	要	要	要	发	发	发	发	发
	大	发	大	发	一	一	大	大	大	大	大	大	一	一	一
	发	大	发	大	大	大	一	一	一	一	一	一	大	大	大
	会	会	会	会	会	会	会	会	会	会	会	会	会	会	会
	民	在	在	在	在	在	在	在	在	在	在	在	在	在	在
	作	作	作	民	民	作	作	作	作	作	作	作	作	作	作
	在	民	民	作	作	民	民	民	民	民	民	民	民	民	民
	全	全	全	展	展	展	展	展	展	展	展	展	展	展	展
	党	展	展	全	全	是	全	全	全	全	全	全	全	全	全
	是	是	是	是	是	全	是	是	平	平	平	平	平	是	平
	展	平	平	平	平	平	平	平	是	是	是	是	是	平	是
	平	党	党	党	新	新	新	新	新	新	新	新	新	新	新
	主	新	新	新	党	党	党	党	党	党	党	党	党	党	党
	新	主	主	主	主	主	主	主	主	主	主	主	主	主	主
熵	9.2702	9.2969	9.3025	9.2959	9.2903	9.2923	9.2882	9.2837	9.2811	9.2792	9.2802	9.281	9.2798	9.2812	9.2817

	1000000	2000000	3000000	4000000	5000000	6000000	7000000	8000000	9000000	10000000	11000000	12000000	13000000	14000000	15000000
概率字	的	的	的	的	的	的	的	的	的	的	的	的	的	的	的
自上而下	一	一	一	一	一	一	一	一	一	一	一	一	一	一	一
概率减小	不	不	不	不	是	了	是	是	是	是	是	是	是	是	是
	是	是	是	是	了	是	了	了	了	了	了	了	了	了	了
	了	了	了	了	不	不	不	不	不	不	不	不	不	不	不
	人	人	人	人	人	这	这	这	这	这	这	这	这	这	这
	子	子	这	这	这	人	人	他	他	他	他	他	他	在	他
	有	道	有	有	有	有	他	在	在	在	在	在	在	他	在
	道	有	在	他	在	在	有	有	有	有	有	有	有	有	有
	这	这	子	在	他	他	在	人	人	人	人	人	就	就	就
	在	在	道	个	就	道	道	道	就	就	就	就	人	人	人
	那	大	就	就	个	就	就	就	道	道	道	道	大	大	大
	大	就	大	大	大	个	个	大	大	大	大	大	我	来	来
	下	下	他	子	道	大	大	个	个	个	我	我	来	我	我
	就	那	个	道	子	上	来	来	来	来	来	来	道	道	道
	上	上	下	上	上	子	那	那	那	那	个	个	个	个	个
	个	个	上	下	下	来	上	上	上	我	那	那	那	那	那
	说	出	那	那	那	我	着	我	我	上	上	上	也	也	也
	你	说	出	也	来	你	我	着	着	也	也	也	上	上	上
	出	你	说	来	也	那	也	也	也	着	着	着	着	着	着
熵	9.65503	9.67128	9.65799	9.6306	9.60856	9.59675	9.57347	9.55483	9.52902	9.50281	9.47738	9.45708	9.44469	9.43154	9.42073

上述是新闻和小说样本最后的熵增趋势和最常出现的 20 个汉字，首先是汉字出现频率的对比，二者出现的汉字有较大差异，人民网上的新闻更偏向于政治，故内容更偏向于政治方面的文字；小说则贴近生活和口语，都是日常容易遇到的汉字，和课件上的常见字应和。

2. 齐夫定律

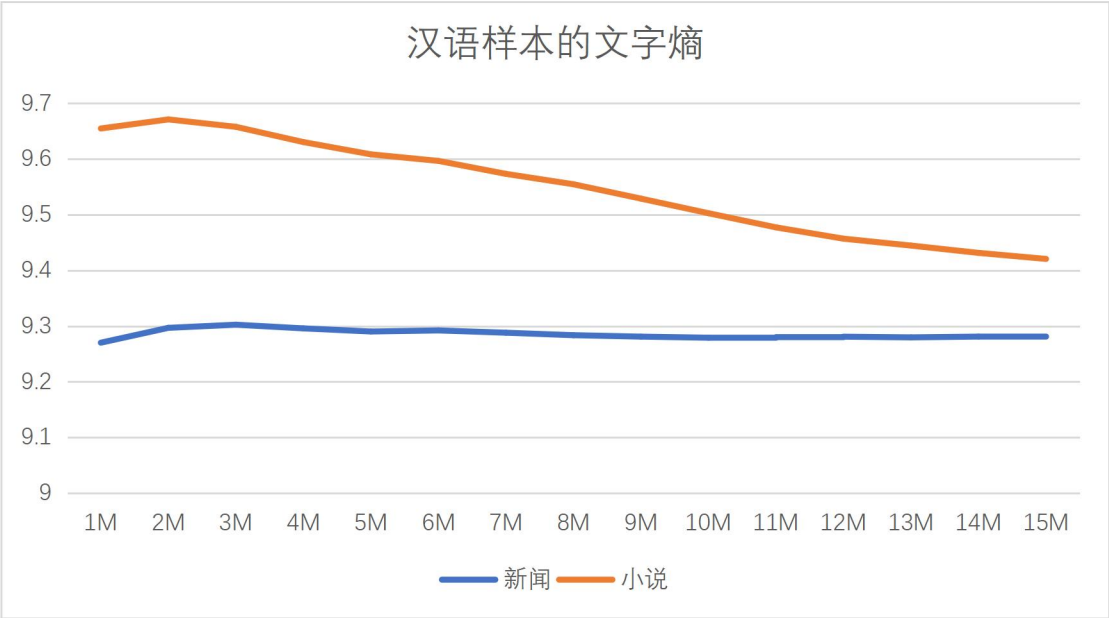
有人在2000万字（14829个词汇）的汉语语料库上进行了统计，频次最高的前10个字（单字词）为：

名次	字（词）	频率	占比(%)	累计占比(%)
1	的	744863	7.7946	7.7946
2	了	130191	1.3624	9.1570
3	在	118823	1.2434	10.4004
4	是	118527	1.2403	11.6407
5	和	83958	0.8786	12.5193
6	一	81119	0.8489	13.3682
7	这	65146	0.6817	14.0499
8	有	53556	0.5604	14.6103
9	他	52912	0.5537	15.1640
10	我	52728	0.5518	15.7158

摘自网站： https://zhuanlan.zhihu.com/p/44646312

说明： 具体结果取决于选取的统计语料类型和规模。 9/107

二者熵的变化趋势对比如下图所示：

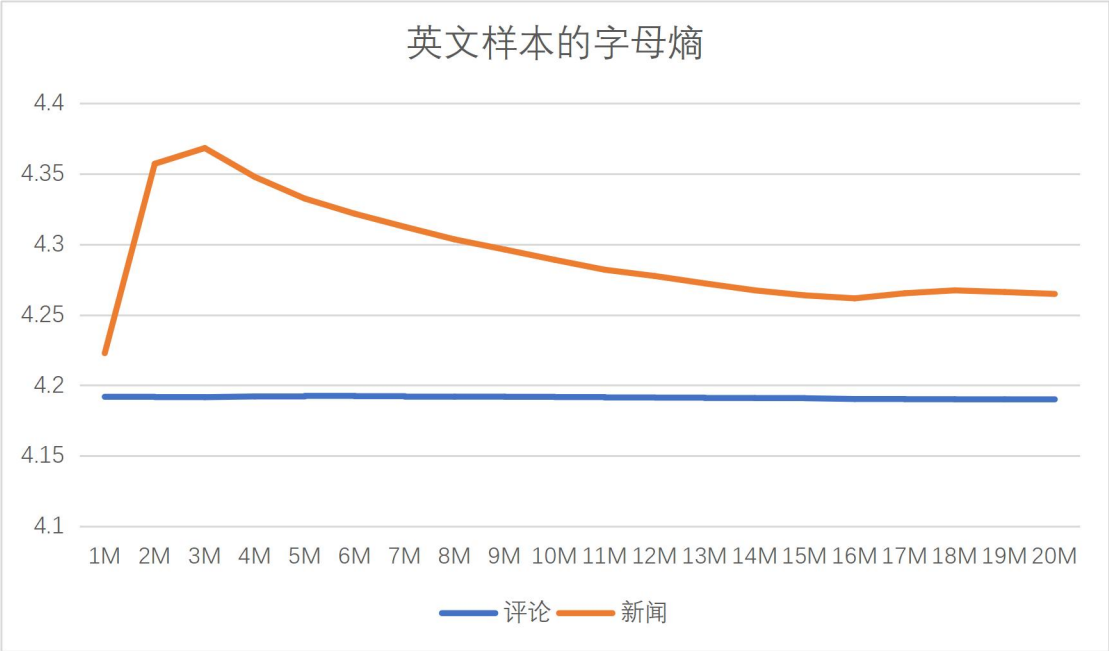


新闻的熵一直趋于稳定，说明这类载体本身的文字变化多样性低于小说，文字结构较为固定；小说的文字熵则有一定的下降趋势，说明随着文字规模扩大，文字出现的可能性趋于稳定。

结合二者的文字熵，我们可以发现新闻的文字熵要低于小说的文字熵，二者在文字规模逐渐增大趋于稳定平滑。这里我的推测是有一些文字是很难在新闻这类正式书面语言里面出现，而会大量出现在小说和日常用语中，故小说的文字类别更多，总熵更大。

2. 字母

英文样本的字母熵图线统计如下所示：



评论的字母熵较为稳定，随着文字规模的扩大变化不明显；新闻的字母熵起初变化较大，但是随着文字规模的扩大，最终趋于稳定。

二者之间的比较我个人认为是没有意义的，这里比较的是字母，是一种没有意义的符号——因为一个字母无法表达出一个具体的意思，且这些样本一定会涵盖所有英文字母。最终的字母熵也只是取决于样本本身，不具有较高探索意义。

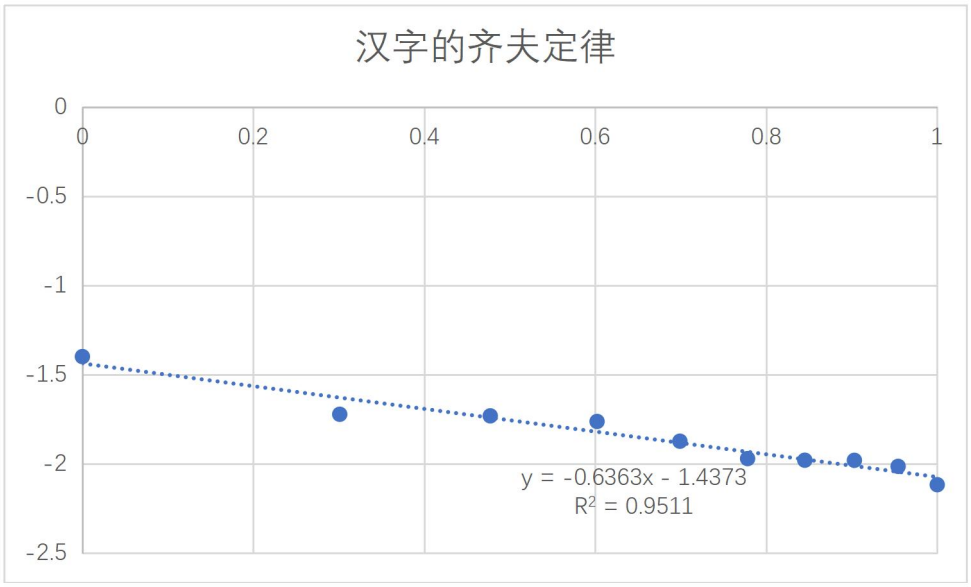
上述是评论数据集中分别取 1M、10M 和 20M 得到的字母概率，发现 E、T、A、

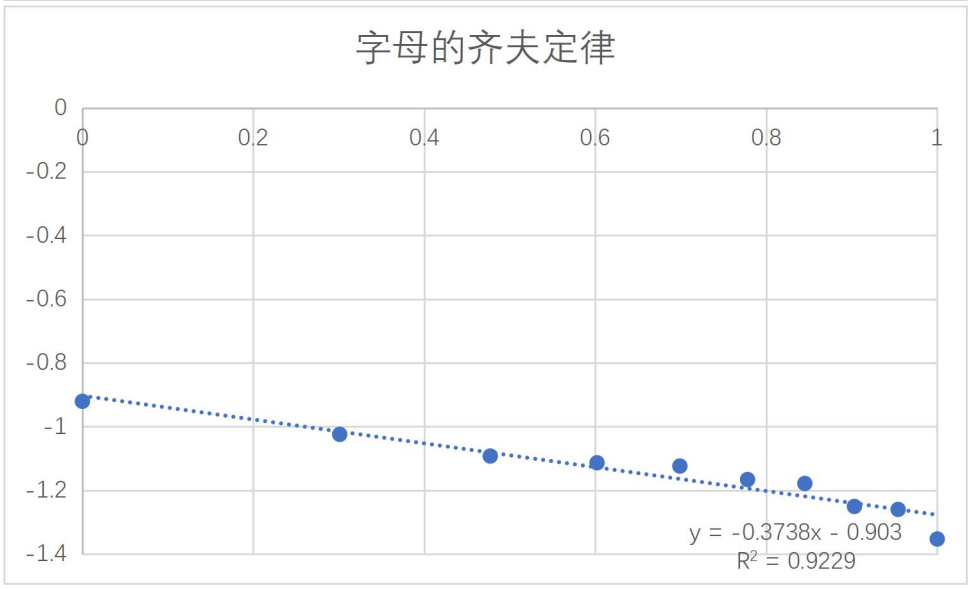
I、O 五个字母出现概率最高，五者概率加和接近占了总字母的一半概率，且随着字母样本数的增多，各自所占比例趋于稳定。

3. 齐夫定律的简单验证

这里利用汉字和字母数据库进行齐夫定律的简单验证，为了使之更直观，这里选择以 rank 排名的对数值作为横坐标，出现概率的对数值作为纵坐标，若两者进行线性回归拟合较好，则说明满足齐夫定律。

这里我只取了前十个最容易出现的汉字和前十个字母进行验证，具体图像如下所示——





4. 对比与总结

上述实验结果中可以得出，汉字和字母规模的增大会导致熵逐渐趋于稳定。这是因为随着文本数量增大，汉字和字母的出现频率趋于稳定。

字母和汉字相比，字母的熵减程度较小，是因为英文字母使用比较随机，故在文本规模增大的情况下，熵不会显著减小。

汉字的熵远高于字母的熵，是因为字母的数目固定且很少，需要多个字母形成单词才能表达出具体含义；而汉字数量远高于二十六个字母，导致其熵比字母熵高。

四、 不足之处

文字规模仍然较小，不足以进行更深入的探究；文字数据比较粗糙，还可以进一步清洗数据；汉字小说中多以第三人称称呼，导致例如“我”等汉字出现频率降低，不符合日常生活中的事实等。

五、 参考文献

1. 利用 urllib 和 BeautifulSoup 爬取维基百科的词条
https://blog.csdn.net/m0_38066258/article/details/77751909?spm=1001.2101.3001.6650.17&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-17-77751909-blog-78462392.pc_relevant_default&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-17-77751909-blog-78462392.pc_relevant_default&utm_relevant_index=18
2. 文字熵的简述
https://blog.csdn.net/jiao_zhoucy/article/details/20362243
3. Python 爬虫入门学习——网页批量爬取文本
https://blog.csdn.net/weixin_54852327/article/details/115916146
4. 一部分代码问题通过 chatgpt-3.5 解决