作业 1：Zipf’s Law 验证与中文信息熵计算

黄家懿 ZY2303203

zy2303203@buaa.edu.cn

Abstract

本报告由两部分组成，第一部分是通过中文语料库验证 Zipf’s Law。第二部分是分别以词和字为单位，使用一元、二元、三元模型计算中文的平均信息熵。

Introduction

验证 Zipf’s Law

- Zipf’s Law（奇普夫定律）

齐普夫定律是美国学者 G.K.齐普夫于 20 世纪 40 年代提出的词频分布定律。它可以表述为：如果把一篇较长文章中每个词出现的频次统计起来，按照高频词在前、低频词在后的递减顺序排列，并用自然数给这些词编上等级序号，即频次最高的词等级为 1，频次次之的等级为 2，…，频次最小的词等级为 D。若用 f 表示频次，r 表示等级序号，则有 fr=C(C 为常数)。

- 停用词

停用词是指在文本处理过程中被忽略或删除的常见词汇。这些词汇通常是频繁出现的功能词或无实际意义的词语，例如介词、连词、冠词、代词等。停用词通常对于文本的含义分析没有太大贡献，且会占据大量的存储空间和计算资源。因此，在文本处理任务（如文本分类、信息检索等）中，常常会预先定义一组停用词，并在处理过程中将它们从文本中移除。

计算中文信息熵

- 信息熵

信息是一个数学上颇为抽象的概念，人们常说信息很多，或者信息很少，但却很难说清

楚信息到底有多少。1948 年，Shannon 指出任何信息都存在冗余，冗余大小与信息中每个符号的出现概率有关。并借鉴热力学的概念，把信息中排除了冗余后的平均信息量称为“信息熵”，同时给出了式（1）所示的计算信息熵的数学表达式。

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{1}$$

- 语言模型

语言模型（Language Model，LM）的作用是估计某一语句在对话中出现的概率。假设 S 表示某一个有意义的句子，由一连串的词 $w_1, w_2, \cdots, w_n$ 排列而成，那么 S 出现的概率是 $w_1, w_2, \cdots, w_n$ 的联合概率，可以表示为式（2）：

$$P(S) = P(w_1, w_2, \cdots, w_n) \tag{2}$$

使用条件概率的公式展开得到式（3）：

$$P(w_1, w_2, \cdots, w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \cdots P(w_n \mid w_1, w_2, \cdots w_{n-1}) \tag{3}$$

其中，$P(w_i \mid w_1, w_2, \cdots w_{i-1})$ 代表前 $i-1$ 个词出现的条件下第 i 个词出现的条件概率。然而，当句子过长时，难以估算 $P(w_i \mid w_1, w_2, \cdots w_{i-1})$，可以通过假设句子具有马尔可夫性来解决。

- 马尔可夫性

如果随机过程的未来状态的条件概率分布(以过去和现在状态为条件)仅取决于前 i 个状态，而不取决于其之前的所有状态，则称随机过程具有马尔可夫性。对于语言模型而言，只需要给出前 i 个词，就可以估计下一个词的概率。

- N-gram 模型

基于马尔可夫性，提出 N-gram 模型，在 N-gram 模型中某一个词的出现金鱼前 N-1 个单词的出现相关。比较常见的有：

unigram 模型： $P(w_1, w_2, \cdots, w_n) = P(w_1)P(w_2)P(w_3) \cdots P(w_n)$

bigram 模型： $P(w_1, w_2, \cdots, w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2) \cdots P(w_n \mid w_{n-1})$

trigram 模型： $P(w_1, w_2, \cdots, w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \cdots P(w_n \mid w_{n-1}, w_{n-2})$

- 信息熵的计算

根据大数定律，如果统计量足够大，字、词、二元词组或三元词组出现的概率大致等于其出现的频率。所以，式中的 $P(x)$ 可近似等于每个字或词在语料库中出现的频率。

二元模型的信息熵计算公式为式（4）：

$$H(X|Y) = -\sum_{x\in X, y\in Y} P(x,y)\log P(x|y) \tag{4}$$

其中，联合概率 $P(x,y)$ 可近似等于每个二元词组在语料库中出现的频率，条件概率 $P(x|y)$ 可近似等于每个二元词组在语料库中出现的频数与以该二元词组的第一个词为词首的二元词组的频数的比值。

三元模型的信息熵计算公式为式（5）所示：

$$H(X|Y,Z) = -\sum_{x\in X, y\in Y, z\in Z} P(x,y,z)\log P(x|y,z) \tag{5}$$

其中，联合概率 $P(x,y,z)$ 可近似等于每个二元词组在语料库中出现的频率，条件概率 $P(x|y,z)$ 可近似等于每个二元词组在语料库中出现的频数与以该二元词组的第一个词为词首的二元词组的频数的比值。

Methodology

M1：验证 Zipf's Law

- 实验环境：

  Python 3.7

  PyCharm Community Edition 2023.1

  jieba 0.42.1

- 实验数据：

  中文语料库：包括《白马啸西风》、《碧血剑》等小说的 txt 格式文档

  标点符号库：cn_punctuation.txt

  中文停词库：cn_stopwords.txt

- 验证 Zipf's Law 的程序基本流程如图 1 所示：

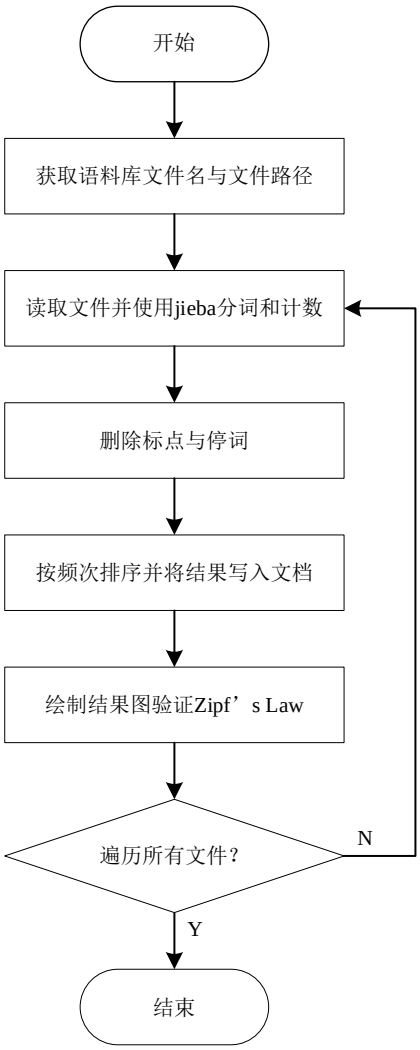


图 1 验证 Zipf’s Law 的程序流程图

- 实验代码：

get_files()：用来获取语料库文件名与文件路径的函数，返回语料库的文件名数组；

words_count(name)：使用 jieba 分词并计数的函数，输入 name 是语料库文件名，返回由分词与分词频数构成的字典；

del_process(counts)：去除标点符号和停词的函数，输入 counts 是由分词与分词频数构成的字典，返回删除标点符号和停词后的新字典；

result_write(count,name)：按频次对分词进行排序并将结果写入结果文档的函数，输入 counts 是由分词与分词频数构成的字典，name 是语料库文件名，返回按序排好的频次数组，便于之后画图；

plot(sort_list,name)：绘制结果图验证 Zipf’s Law 的函数，输入 sort_list 是按序排好的频次数组，name 是语料库文件名，无返回值。

M2：计算中文信息熵

- 实验环境：

  Python 3.7

  PyCharm Community Edition 2023.1

  jieba 0.42.1

- 实验数据：

  中文语料库：包括《白马啸西风》、《碧血剑》等小说的 txt 格式文档

  标点符号库：cn_punctuation.txt

  中文停词库：cn_stopwords.txt

- 计算中文信息熵的程序基本流程图如图 2 所示：

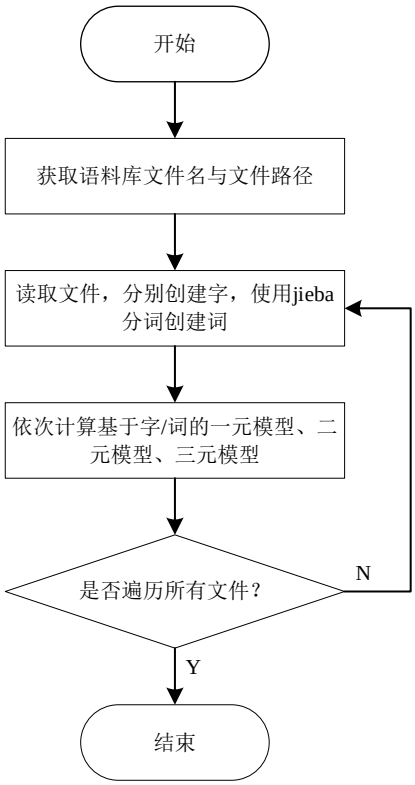


图 2 计算中文信息熵的程序流程图

- 实验代码：

  get_files()：用来获取语料库文件名与文件路径的函数，返回语料库的文件名数组；

  word_genaration(name)：由读取的文本文档生成字和词的函数，输入 name 是语料库文件名，删除标点符号和停词后分别返回字和词各自的数组；

  get_unigram_tf(words)，get_bigram_tf(words), get_trigram_tf(words)：分别是用来构建一元、二元、三元模型的函数，输入 words 是字或词的数组，返回好的一元、二元、三元模

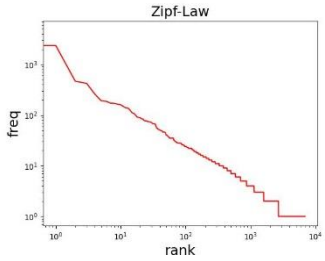
型；

char_calculate1(word,is_ci,name)，char_calculate2(word,is_ci,name)，char_calculate3(word,is_ci,name)：分别是用来计算一元、二元、三元模型下信息熵的函数，输入 word 是字或词的数组，is_ci 代表输入是基于字还是基于词，name 是语料库文件名，返回信息熵。

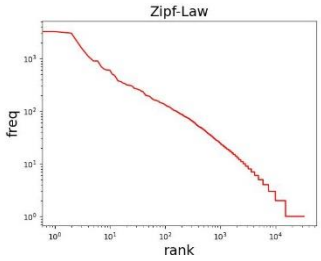
Experimental Studies

验证 Zipf's Law

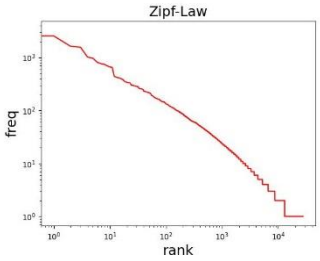
使用 16 本金庸小说进行实验，图 3 所示式实验结果，其中，横轴是分词的排序等级，纵轴是分词的出现频次，由于在代码中对横纵坐标都进行了对数处理，因此 Zipf's Law 中提出的排序等级与出现频次乘积为常数结论在对数处理后应表现为二者之和为常数，即结果图应大致呈一条线形直线。显然结果图与理论分析基本吻合，因此，成功论证了 Zipf's Law。



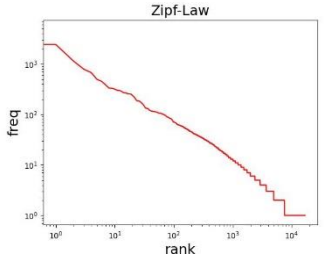
a.《白马啸西风》



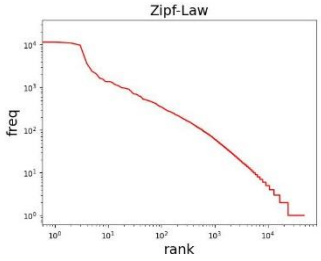
b.《碧血剑》



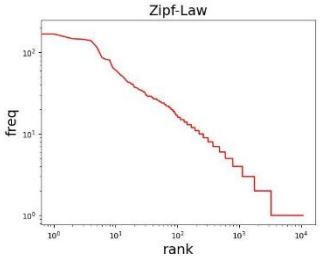
c.《飞狐外传》



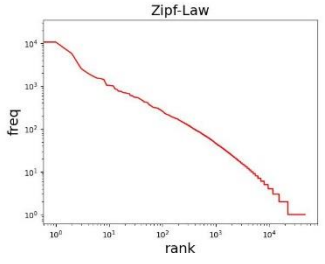
d.《连城诀》

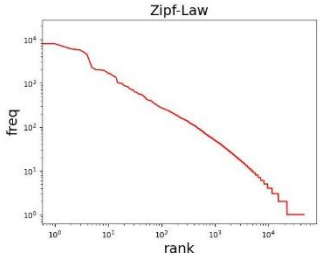


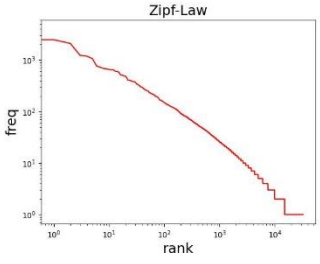
e.《鹿鼎记》



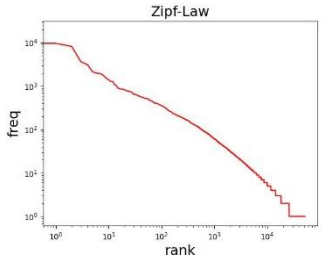
f.《三十三剑客图》

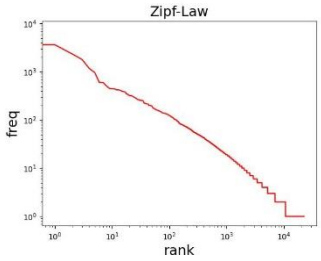


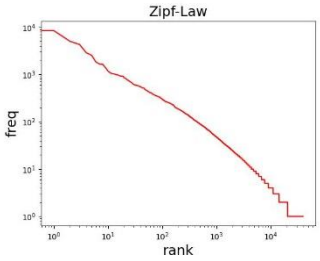




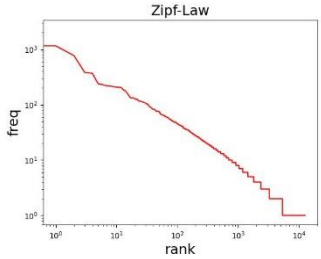
g.《射雕英雄传》　　h.《神雕侠侣》　　i.《书剑恩仇录》

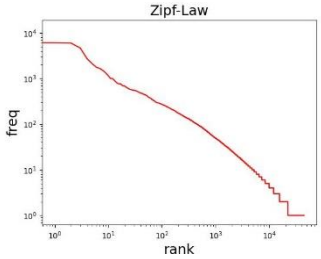


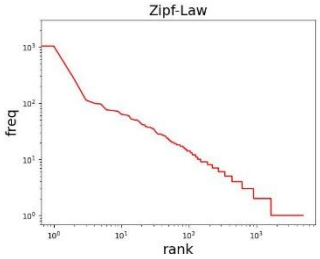




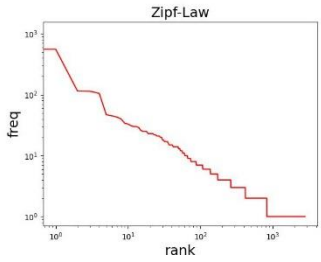
j.《天龙八部》　　k.《侠客行》　　l.《笑傲江湖》







m.《雪山飞狐》　　n.《倚天屠龙记》　　o.《鸳鸯刀》



p.《越女剑》

图 3 Zipf's Law 结果图

计算中文信息熵

对金庸的 16 本小说进行了中文信息熵的计算，表 1 所示是各语料库基于字的信息熵计算结果。

表 1 基于字的中文信息熵计算结果

	一元模型（比特/字）	二元模型（比特/字）	三元模型（比特/字）
《白马啸西风》	9.2322	4.0851	1.2095
《碧血剑》	9.7573	5.6740	1.7948
《飞狐外传》	9.6309	5.5684	1.8649
《连城诀》	9.5163	5.0897	1.6390

《鹿鼎记》	9.6606	6.0188	2.4088
《三十三剑客图》	10.0122	4.2818	0.6504
《射雕英雄传》	9.7533	5.9651	2.1956
《神雕侠侣》	9.5585	6.0551	2.3738
《书剑恩仇录》	9.7597	5.5963	1.8614
《天龙八部》	9.7843	6.1142	2.3513
《侠客行》	9.4382	5.3770	1.8188
《笑傲江湖》	9.5176	5.8549	2.3610
《雪山飞狐》	9.5054	4.7993	1.3020
《倚天屠龙记》	9.7077	5.9825	2.2758
《鸳鸯刀》	9.2210	3.6511	0.8936
《越女剑》	8.8082	3.0944	0.8367

表 2 所示是各语料库基于词的信息熵计算结果。

表 2 基于词的中文信息熵计算结果

	一元模型 （比特/词）	二元模型 （比特/词）	三元模型 （比特/词）
《白马啸西风》	11.1311	2.9089	0.3699
《碧血剑》	12.8846	3.9618	0.4326
《飞狐外传》	12.6261	4.0404	0.4612
《连城诀》	12.2068	3.5889	0.3692
《鹿鼎记》	12.6311	4.9924	0.8401
《三十三剑客图》	12.5292	1.8181	0.0911
《射雕英雄传》	13.0353	2.2196	0.5387
《神雕侠侣》	12.4250	4.9382	0.7731
《书剑恩仇录》	12.7274	4.1355	0.4987
《天龙八部》	13.0183	4.8385	0.6647
《侠客行》	12.2875	3.9897	0.5135
《笑傲江湖》	12.5237	4.8373	0.7963
《雪山飞狐》	12.0585	3.0641	0.2908

《倚天屠龙记》	12.8911	4.6843	0.6445
《鸳鸯刀》	11.1030	2.1524	0.2428
《越女剑》	10.4762	1.7331	0.2440

通过对比同一语料库的一元、二元、三元模型，可以看到，随着 N 的取值增大，基于字/词的信息熵减小，这是因为 N 越大，组成该组字/词的字/词数量越多，内容越复杂和准确，因此不确定性小，意味着熵小。

通过对比同一语料库在基于字和基于词的信息熵计算结果，可以看到，在一元模型下，基于字的信息熵小于基于词的信息熵，这可能是因为一元模型不考虑前后文，有意义的词重复率低，无意义的字重复率较高，进而使得基于字的熵小于基于词的熵；在二元和三元模型下，基于字的信息熵大于基于词的信息熵，这可能是因为基于词在二元和三元的划分下能更好地考虑了基本的语法结构，例如形容词+名词等组合，因此不确定性更低，熵更小。

Conclusions

验证 Zipf's Law

- 结果图中，横轴代表分词的排序等级，纵轴代表分词的出现频次，经过对数处理后大致呈一条直线，验证了 Zipf's Law 的正确性。

计算中文信息熵

- 随着 N 的取值增大，基于字/词的信息熵减小。这是因为 N 越大，内容越复杂和准确，因此不确定性小，意味着熵小；
- 在一元模型下，基于字的信息熵小于基于词的信息熵，这可能是因为一元模型不考虑前后文，有意义的词重复率低，无意义的字重复率较高，进而使得基于字的熵小于基于词的熵；
- 在二元和三元模型下，基于字的信息熵大于基于词的信息熵，这可能是因为基于词在二元和三元的划分下能更好地考虑了基本的语法结构，例如形容词+名词等组合，因此不确定性更低，熵更小。

References

[1] Brown P F, Della Pietra S A, Della Pietra V J, et al. An estimate of an upper bound for the entropy of English[J]. Computational Linguistics, 1992, 18(1): 31-40.

[2] https://blog.csdn.net/weixin_43353612/article/details/105147148

[3] https://zhuanlan.zhihu.com/p/658563402