NLP 作业

——在中文语料库上验证齐夫定律，计算中文的平均信息熵

院（系）名称	自动化科学与电气工程学院
学生姓名	王海滨
学生学号	ZY2303708

2024 年 4 月 1 日

一、引言

齐夫定律，由美国语言学家乔治·齐夫在20世纪40年代提出，是描述语言中词汇分布的一种规律，指出在任何给定语料库中，词的频率与其频率排名成反比。这一发现揭示了语言使用中的一种普遍现象：少数高频词汇在交流中被频繁使用，而大多数词汇使用频率低。齐夫定律的普适性不仅对语言学有重要影响，也在信息理论等多个领域内有着广泛应用。

选用中文语料库验证齐夫定律，旨在探索这一定律在不同语言体系中的适用性。中文的独特性，如书写系统和语法结构，提供了一个独特的视角来观察和理解齐夫定律。此外，这一研究有助于深入分析中文的语言特性如何影响词汇的分布与使用，对提高中文文本处理技术和语言模型的准确性及效率具有实际意义。

此外，引入信息熵概念，量化语言的不确定性，可以进一步理解齐夫定律在中文语料库中的表现。信息熵反映了语言的多样性和复杂性，而齐夫定律所揭示的词频分布特性对语言的信息熵有直接影响。分析中文语料库中的词频分布与信息熵之间的关系，不仅能够深化我们对中文交流效率和表达丰富性平衡的理解，也为比较不同语言的信息处理效率提供了新的视角。

二、研究方法(齐夫定律）

语料库的选择：本研究采用的中文语料库由金庸撰写的十六部武侠小说组成。重要的是要指出，尽管这些小说中的语言风格与现代标准中文存在显著差异，但武侠小说中的表达方式本身可视为一种独特的语言风格。此外，这种表达方式不仅反映了特定的文化背景和时代特色，也体现了金庸个人的语言特点。因此，虽然研究对象具有其特殊性，但它为探讨齐夫定律在特定文体和作者风格下的适用性提供了一个有趣且具有代表性的案例。

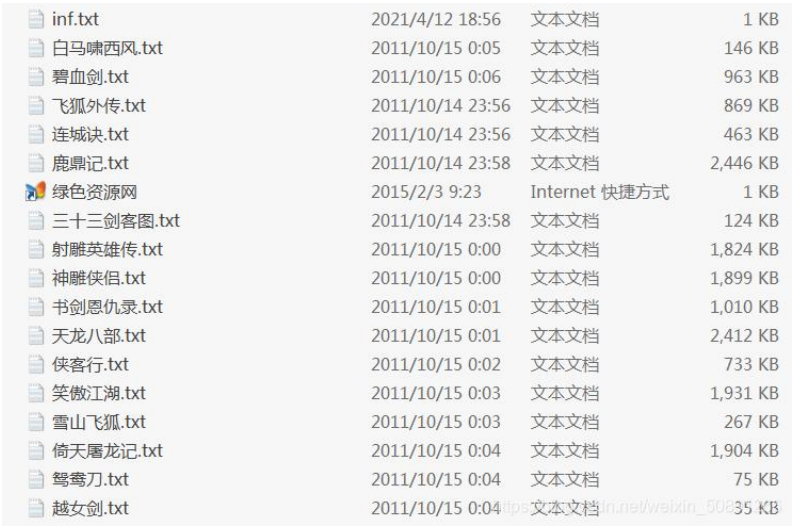


图1. 本次研究所使用的语料库

数据处理方法：本研究所依赖的数据集包括金庸先生创作的16

部小说。鉴于原始文本中含有大量的乱码、无效内容以及重复的中英文符号，对数据集进行彻底的预处理变得尤为重要。预处理步骤包括：首先，去除所有隐藏的字符以清理文本；其次，删除所有非中文字符，确保分析的纯净度；最后，在不考虑上下文的情况下，移除所有标点符号，避免对分词结果产生干扰。

本研究采用了 jieba 分词库进行文本处理，jieba 是 Python 中广泛使用的一个中文分词工具。在此实验中，我们采用了 jieba 的精确模式进行分词，旨在最大程度上保证文本分词的准确性和效率。这一系列预处理措施为后续的数据分析提供了干净、可靠的文本基础。

```
#预处理文本
def getCorpus( rootDir):
    corpus = []
    r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~]+'
    for file in os.listdir(rootDir):
        path  = os.path.join(rootDir, file)
        if os.path.isfile(path):
            with open(os.path.abspath(path), "r", encoding='gbk',errors='ignore') as file:
                filecontext = file.read();
                filecontext = re.sub(r1, '', filecontext)
                filecontext = filecontext.replace("\n", '')
                filecontext = filecontext.replace(" ", '')
                seg_list = jieba.cut(filecontext)
                corpus += seg_list
    return corpus
```

预处理代码

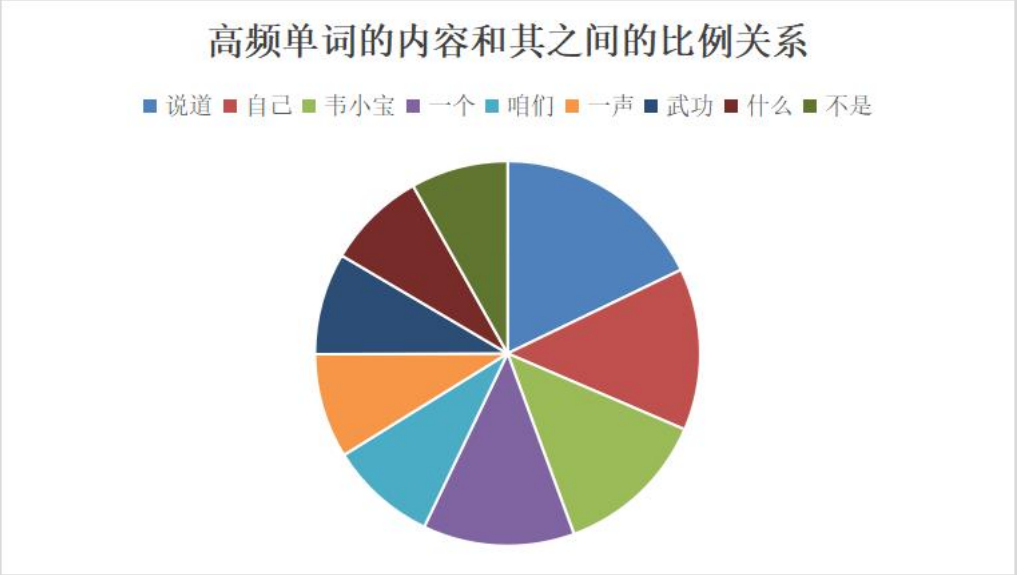
数据处理方法：本研究采用的统计方法直接且高效，基于 Python 编程语言实现。具体而言，分析过程首先利用 jieba 分词库处理文本，将整个文本分解成独立的词汇单元。分词后，我们通过 Python 的 count 函数来实现词频统计，这一步骤紧随停用词的筛选过程，后者负责移除文本中的常见但信息量较低的词汇（如“的”、“是”、“在”等），以减少对统计结果的噪声干扰。完成这些步骤之后，我们更新词频统计结果，专注于提取出现频率最高的前 30 个词汇，并对这些词汇的频次进行详细统计分析。此方法不仅确保了统计过程的简洁性，也提高了数据处理和分析的准确性，为探究齐夫定律在所选中文语料库中的应用提供了坚实的基础。

```
def process_text(words, stopwords):
    """处理文本：分词、过滤停用词和单字"""
    return [word for word in words if word not in stopwords and len(word) > 1]
```

停用词筛选代码

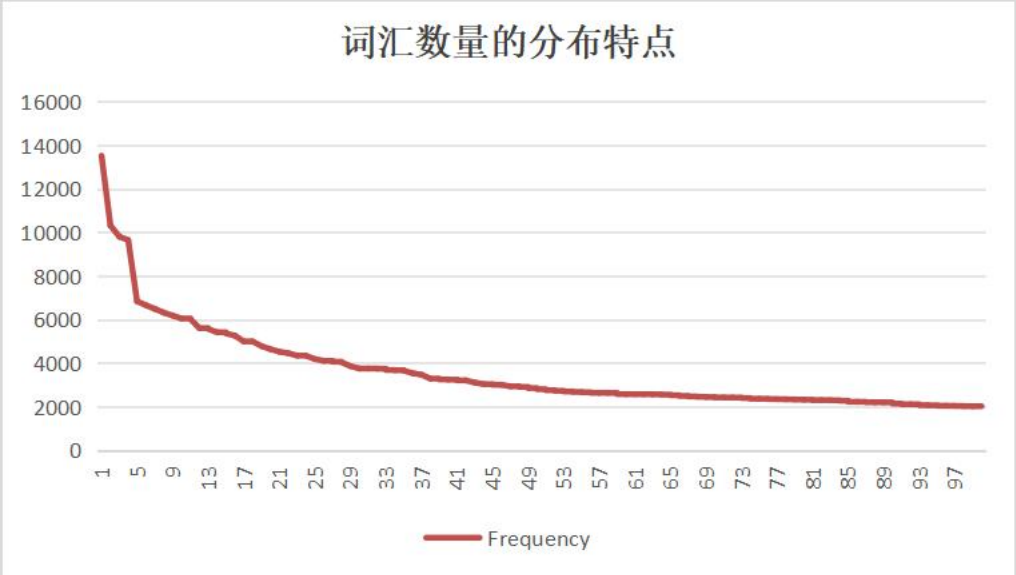
三、结果分析(齐夫定律）

初步观察：我们首先对词频排名前十的词汇进行了分析，探讨了这些高频词汇的具体内容及其相互之间的比例关系。通过这一步骤，我们旨在揭示最常用词汇的分布特点，为进一步的分析奠定基础。



测试结果的可视化

词频可视化:为了深入理解词频分布情况，并有效验证齐夫定律，我们对排名前 100 的词汇进行了词频可视化处理。这一过程不仅使得数据结果直观易懂，也便于观察词汇使用的整体趋势和模式。



测试结果的可视化

从可视化结果来看，我们的发现并未完全符合齐夫定律的预期。分析这一现象，我们认为主要原因有两点：首先，所选取的语料库规模可能尚未达到足够大的量级，导致虽有大致的趋势呈现，但无法精确地对应齐夫定律的理论模型。其次，停用词的筛选可能也对结果产生了影响。考虑到停用词列表中包含了许多高频的日常用语，其筛除可能在一定程度上扭曲了词频的真实分布情况。

四、结论(齐夫定律）

本研究利用金庸创作的十六部武侠小说作为中文语料库，旨在探究齐夫定律在中文文本中的适用性。经过严格的数据预处理和利用jieba分词库进行文本分析后，我们对高频词汇进行了统计和词频可视化分析。

分析结果显示，虽然词频分布与齐夫定律的理论预期存在一定偏差，但这一现象为理解齐夫定律在特定中文文本中的适用性提供了有价值的见解。我们认为，主要偏差原因包括语料库规模的限制和停用词筛选策略的影响。这些发现强调了进行语言学研究时选择适当的语料库规模和优化数据处理策略的重要性。

尽管未能完全证实齐夫定律在所有情况下的普遍适用性，本研究为未来在更广泛语料库和不同文本类型中验证齐夫定律提供了方法和参考。未来研究可通过扩大语料库规模和优化数据处理方法，进一步探索齐夫定律的适用范围和限制。

五、研究方法(中文信息熵计算)

对于中文(分别以词和字为单位）的平均信息熵的计算，我们的数据预处理方面跟齐夫定律时稍有不同，即不进行停用词的限制，其余均一致。

以下为以字计算中文平均信息熵的代码作为例子：

```
def get_tf(tf_dic, words):
# 词频统计，方便计算信息熵
    for i in range(len(words)-1):
        tf_dic[words[i]] = tf_dic.get(words[i], 0) + 1


def cal_unigram(corpus,count):
    before = time.time()
    split_words = []
    words_len = 0
    line_count = 0
    words_tf = {}
    for line in corpus:
        for x in jieba.cut(line):
            split_words.append(x)
            words_len += 1
        get_tf(words_tf, split_words)
        split_words = []
        line_count += 1

    print("语料库字数:", count)
    print("分词个数:", words_len)
    print("平均词长:", round(count / words_len, 5))
```

```
    entropy = []
    for uni_word in words_tf.items():
        entropy.append(-(uni_word[1] / words_len) * math.log(uni_word[1] / words_len, 2))
    print("基于词的一元模型的中文信息熵为:", round(sum(entropy), 5), "比特/词")
    after = time.time()
    print("运行时间:", round(after - before, 5), "s")
```

六、结果分析(中文信息熵计算）

在分析中文信息熵的结果时，我们得到了一元、二元、和三元模型的信息熵分别为 12.01086 比特/词、6.89278 比特/词和 2.41723 比特/词。这些结果揭示了几个关键的研究发现，同时与英文的信息熵进行比较，可以体现出中文的独特特点。

分词模型	语料字数	分词个数	平均词长	信息熵/词
一元（字）	7419881	4430235	1.67483	12.01086
二元	7419881	4430235	1.67483	6.89278
三元	7419881	4430235	1.67483	2.41723

信息熵的计算结果

中文信息熵的特点

一元模型信息熵较高：结果显示中文在一元模型下的信息熵达到了 12.01086 比特/词，反映出单个汉字在语言中承载的信息量非常大。这与汉字作为意音文字的特性有关，每个字不仅表示特定的声音，还蕴含着特定的意义。

信息熵随模型复杂度降低：在二元和三元模型中，信息熵分别降至 6.89278 比特/词和 2.41723 比特/词。这表明，随着上下文信息的增加，中文文本的不确定性显著减少，反映出中文在词汇搭配和句式构造上的规律性和约束性。

与英文的对比

中英文信息熵差异：中文的一元模型信息熵通常高于英文，这反映了中文字符比英文单词携带更多的信息。然而，在更复杂的模型中，中文信息熵的降低速度快于英文，可能由于中文具有更多固定搭配和成语，使得语言的可预测性增强。

语言结构影响：中文的平均词长较短，与其作为意音文字的特性相关。这种结构差异导致中英文在信息熵上的表现不同，英文依赖于连续的字符形成单词，而中文每个汉字本身就能独立承载意义。

七、结论(中文信息熵计算)

本研究通过计算中文文本的一元、二元、和三元信息熵，深入探讨了中文语言的信息分布特性及其独特性。结果表明，随着模型复杂度的增加，中文信息熵显著下降，从一元模型的 12.01086 比特/词到三元模型的 2.41723 比特/词，这一变化反映了中文在词与词组合时的高度规律性和上下文约束性。与英文相比，中文展现出更高的单字信息量和更快的信息熵下降速度，凸显了其作为一种高度结构化语言的特点。

这一发现对于理解中文的语言结构、优化中文文本处理技术具有重要意义。它不仅验证了中文信息处理的独特效率，也为未来基于中文的自然语言处理技术的发展提供了重要的理论支持和实证基础。通过本研究，我们得以更清晰地认识到在设计和应用中文处理算法时，需要充分考虑到语言的这些独特性质，以实现更高效和精准的信息理解与传递。

参考资料：

1. python 读取 txt 文件中文-python 读取中文 txt 文本的方法-CSDN 博客
2. 齐普夫定律 Zipf's law - 集智百科 - 复杂系统|人工智能|复杂科学|复杂网络|自组织 (swarma.org)
3. jieba · PyPI
4. [Python] 信息论：计算机自信息，信息熵，对比中文和英文信息熵。_分别收集尽量多的英语和汉语文本,编写程序计算这些文本中英语字母和汉字的熵,对比-CSDN 博客
5. python 报错 UnicodeDecodeError: 'gbk ' codec can 't decode byte, 文本乱码解决方法 unicodedecodeerror:'gbk' codec can't decode byte -CSDN 博客
6. Python——jieba 优秀的中文分词库（基础知识+实例）-CSDN 博客