Report of Homework One

张海尼
ZY2303215

Abstract

报告主要包括两个部分。第一部分：通过中文语料库验证 Zipf's Law；第二部分：阅读 Entropy Of English，计算中文（分别以词和字为单位）的平均信息熵。

Introduction

I1: Zipf's Law

Zipf's Law 为语言学家 George Kingsley Zipf 总结的经验定律。Zipf's Law 指出，在大型自然语言语料库中，单词的频率与其在频率表中的排名成反比，频率与排名有近似幂律关系。

I2: 平均信息熵

信息熵用于衡量一段信息中包含的平均信息量或不确定性。信息的不确定性越高，则信息熵越大，反之则越小。

信源发送信息的不确定性使用符号出现的概率进行度量。不确定性函数 f 是概率 P 的减函数，相互独立的符号的不确定性满足 $f\left(P_1,P_2\right)=f\left(P_1\right)+f\left(P_2\right)$，同时满足这两个条件的 f 如下：

$$f(P)=-log\ p$$

计算信源的平均不确定性时，需要考虑该信源中所有符号的不确定性（不同符号相互独立），此时得到信息熵的计算式如下：

$$H(X)=E[-log\ p_i]=-\sum_{i=1}^{n}p_i log\ p_i$$

其中，p_i 为不同符号出现的概率，n 为总符号数。log 一般以 2 为底，单位为比特。

Methodology

语言模型——N-Gram

统计语言模型是以统计学位基础的语言模型，它基于预先收集的大规模语料数据，以人类语言习惯为标准，预测不同文本序列在语料库中可能出现的概率，以此评估文本序列符合语言习惯的程度。

假定 S 表示一个有意义的句子，由一串以特定顺序排列的词 $w_1, w_2, \ldots, w_n$ 组成，n 为句子长度。此时 S 在文本中出现的可能性如下：

$$P(S) = P(w_1, w_1, w_2, \ldots, w_{i-1}) = P(w_1)P(w_2 \mid w_1)\ldots P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

其中，$P(w_1)$ 表示第一个词为 w_1 的概率，$P(w_2 \mid w_1)$ 表示在已知第一个词为 w_1 的前提下，第二个词为 w_2 的概率，以此类推。

当句子过长时，$P(w_i \mid w_1, w_2, \ldots, w_{i-1})$ 难以估算，因此引入马尔科夫假设：一个词出现的概率只与它前面出现的 k 个词有关，得到条件概率的简化式如下：

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

当 $k = 1$ 时，每个词出现的概率与其他词无关，为一元模型，此时信息熵的计算式如下：

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

当 $k = 2$ 时，为二元模型，此时信息熵的计算式如下：

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

当 $k = 3$ 时，为三元模型，此时信息熵的计算式如下：

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

Experimental Studies

E1: 验证 Zipf's Law

1.1 数据处理与验证的算法流程

1) 所用数据库为 10 部金庸小说；
2) 进行数据预处理，删除部分文件开头的无用信息；
3) 使用 jieba 算法库完成分词；
4) 进行词频统计；

5) 删除统计结果中的无意义符号，包括标点符号和空白符号等；
6) 绘图验证 Zipf’s Law。

1.2 部分代码

```
# 去除无用信息
txt = txt.replace('本书来自www.cr173.com免费txt小说下载站\n更多更新免费电子书请关注www.cr173.com', '')

# 分词
words = jieba.lcut(txt)

# 统计词频
counts = {}
for word in words:
    counts[word] = counts.get(word, 0) + 1

# 去除无意义字符
for word in extra_characters:
    if word in counts:
        del counts[word]

# 排序
items = list(counts.items())
items.sort(key=lambda x: x[1], reverse=True)
sort_list = sorted(counts.values(), reverse=True)

# 验证zipf-law
figPath = 'cut_word_fig/' + name + ".jpg"
x = [i for i in range(len(sort_list))]
plt.plot(x, sort_list)
```

1.3 实验结果

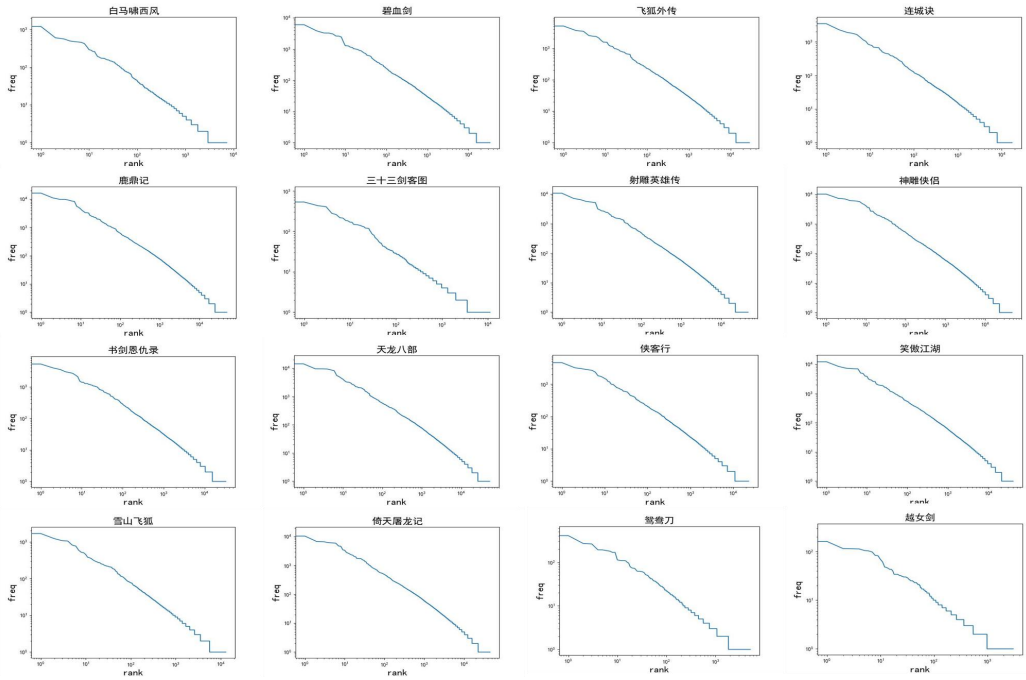

Figure 1：不同输入下的验证结果图

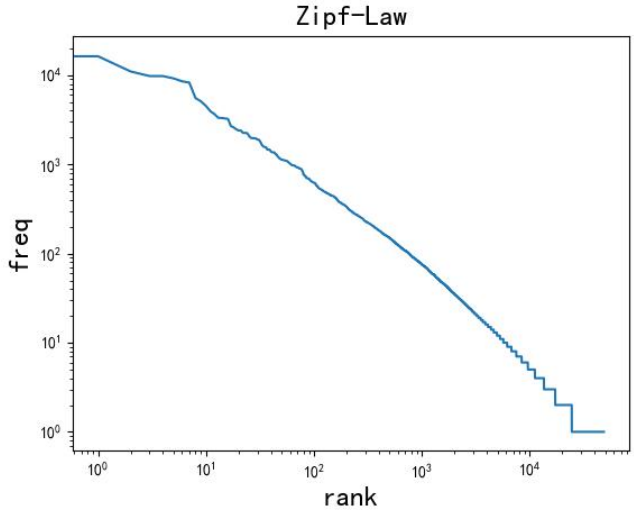

Figure 1：总体验证结果图

E2：计算平均信息熵

2.1 计算信息熵的算法流程

1) 所用数据库为 10 部金庸小说；
2) 进行数据预处理，删除部分文件开头的无用信息；
3) 删除词频表中的无意义符号及中文停词；
4) 计算不同语言模型的词频表；
5) 计算不同语言模型下的平均信息熵。

2.2 语言模型部分代码

2.2.1 一元模型

```
def entropyUnigram(counts):
    word_len = sum([item[1] for item in counts.items()])

    entropy = 0
    for word in counts.items():
        entropy -= word[1]/word_len * math.log(word[1] / word_len, 2)

    return entropy
```

2.2.2 二元模型

```python
def entropyBigram(uiCounts, biCounts):
    word_len = sum([item[1] for item in biCounts.items()])

    entropy = 0
    for word in biCounts.items():
        p_xy = word[1]/word_len
        p_x_y = word[1]/uiCounts[word[0][0]]
        entropy -= p_xy * math.log(p_x_y, 2)

    return entropy
```

2.2.3 三元模型

```python
def entropyTrigram(biCounts, triCounts):
    word_len = sum([item[1] for item in triCounts.items()])

    entropy = 0
    for word in triCounts.items():
        p_xy = word[1]/word_len
        p_x_y = word[1]/biCounts[(word[0][0], word[0][1])]
        entropy -= p_xy * math.log(p_x_y, 2)

    return entropy
```

2.3 实验结果

Table 1: 信息熵实验结果

	一元字	二元字	三元字	一元词	二元词	三元词
白马啸西风	9.225088	4.088662	1.211628	11.19527	2.879637	0.354012
碧血剑	9.754924	5.6754	1.795511	12.88506	3.962157	0.430718
飞狐外传	9.630064	5.569083	1.864978	12.62592	4.040443	0.460867
连城诀	9.515225	5.090152	1.638962	12.20663	3.589008	0.368527
鹿鼎记	9.658505	6.019911	2.409586	12.63926	4.992722	0.83437
三十三剑客图	10.0067	4.28446	0.650668	12.53354	1.809615	0.091192
射雕英雄传	9.751662	5.965195	2.1961	13.04546	4.59377	0.533156
神雕侠侣	9.662769	6.002454	2.282799	12.76041	4.687411	0.626518
书剑恩仇录	9.75771	5.598028	1.8616	12.72734	4.687411	0.497441
天龙八部	9.782239	6.114913	2.351673	13.01884	4.838824	0.663293
侠客行	9.434942	5.380029	1.819657	12.28746	3.992884	0.512686
笑傲江湖	9.51581	5.856499	2.361396	12.52379	4.838776	0.795459
雪山飞狐	9.50104	4.801486	1.303221	12.05641	3.065894	0.290599
倚天屠龙记	9.706427	5.983031	2.276121	12.8935	4.685251	0.642583
鸳鸯刀	9.210769	3.656518	0.896111	11.1362	2.146505	0.232316
越女剑	8.782444	3.109167	0.842115	10.50206	1.73514	0.233117
平均	9.55602	5.199687	1.735133	12.31482	3.784091	0.472928

Conclusions

C1: 验证 Zipf's Law

由各小说集以及总体的词频统计结果可知，词频的排名（降序排列）与其频率的对数存在近似反比的关系，且随着语料库规模的增大，曲线的平滑程度更大，验证了 Zipf's Law。

C2: 计算信息熵

由实验结果可知，使用一元模型、二元模型、三元模型计算信息熵时，基于字和词的统计结果均有相似的规律，即 k 值越大，信息熵越小。当 k 值越大时，用于计算信息熵的单位词组越多，所涵盖的信息量越大，信息的不确定性越小，因此信息熵越小。

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.