北京航空航天大學
BEIHANG UNIVERSITY

深度学习与自然语言处理

学 院 名 称	自动化科学与电气工程学院
专 业 名 称	自动化
学 生 姓 名	张一凡
指 导 教 师	秦曾昌

2023 年 3 月

深度学习与自然语言处理第一次大作业-中文语料库平均信息熵计算

一、实验内容

参考 Entropy_of_English_PeterBrown 这篇文章来计算中文语料库的平均信息熵

二、实验数据

实验数据集包括金庸的 16 部武侠小说，文件格式为.txt

实验步骤

首先对于实验数据进行预处理，因为实验要求我们计算的是中文字符的平均信息熵，因此我们应该将除中文字符之外的字符（如英文字符、标点符号、噪声等）进行剔除。这里我们使用到的是正则表达式来进行提取，将\u4E00-\u9FA5 之间的字符进行提取则为我们需要的中文字符，同时将英文字符（[a-z A-Z 0-9]）以及标点符号剔除。将处理后的文本每一行读取出来成为一个自然段进行存储，最后存入一个.txt 文件便于数据的读入。需要注意的是，对于噪声数据，如一些出现频次极低的字符，我们可判定为噪声字符，这里我们直接替换成空字符，并在结果中将我们替换的噪声数据进行列举。

其次，是我们通过阅读信息熵这篇文献所获得的如何计算信息熵，定义公式为

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

三、实验算法设计

我们采用上述信息熵的计算公式，通过 N-Gram 这一语言模型对文本序列进行划分。该语言模型就是用来计算一个句子的概率，判断这句话是否被认定为正常句子的概率，

给定一个词语序列：$S = W_1, W_2, ..., W_K$，则它的概率可以表示为：

$$\mathrm{P(S)} = \mathrm{P}(W_1, W_2, \ldots, W_K) = p(W_1) P(W_2 |\ W_1) \ldots \mathrm{P}(W_K |\ W_1, W_2, \ldots, W_{k-1})$$

将上面的链式规则计算 P(W)可以写为：$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i |\ w_1 w_2 ... w_{i-1})$

但是事实上，如果用以上模型来计算条件概率会出现参数空间过大，计算资源严重不足以及数据稀疏严重无法提供有效的信息熵计算等问题。因而我们采用 N-Gram 语言模型，引入马尔科夫假设，随意一个词语出现的概率只与它前面的有限的一个或者几个词语有关，上面的计算方式是通过马尔科夫假设得出的，马尔科夫假设是指假设第 w_i 个词语只与它前面的 k 个词语相关，这样我们就得到前面的条件概率计算简化如下：

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

这样计算的 P(W)简化如下：

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i |\ w_{i-k} \ldots w_{i-1})$$

当 k=0 时，这个时候对应的模型叫做一元模型，即 w_i 与它前面的 0 个词相关，即 w_i 不与任何词相关，每个词语都是相互独立的，P(W)计算如下：

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i)$$

当 k=1 时，对应的模型叫做二元模型，此时 w_i 与它前面一个词相关，P(W)计算如下：

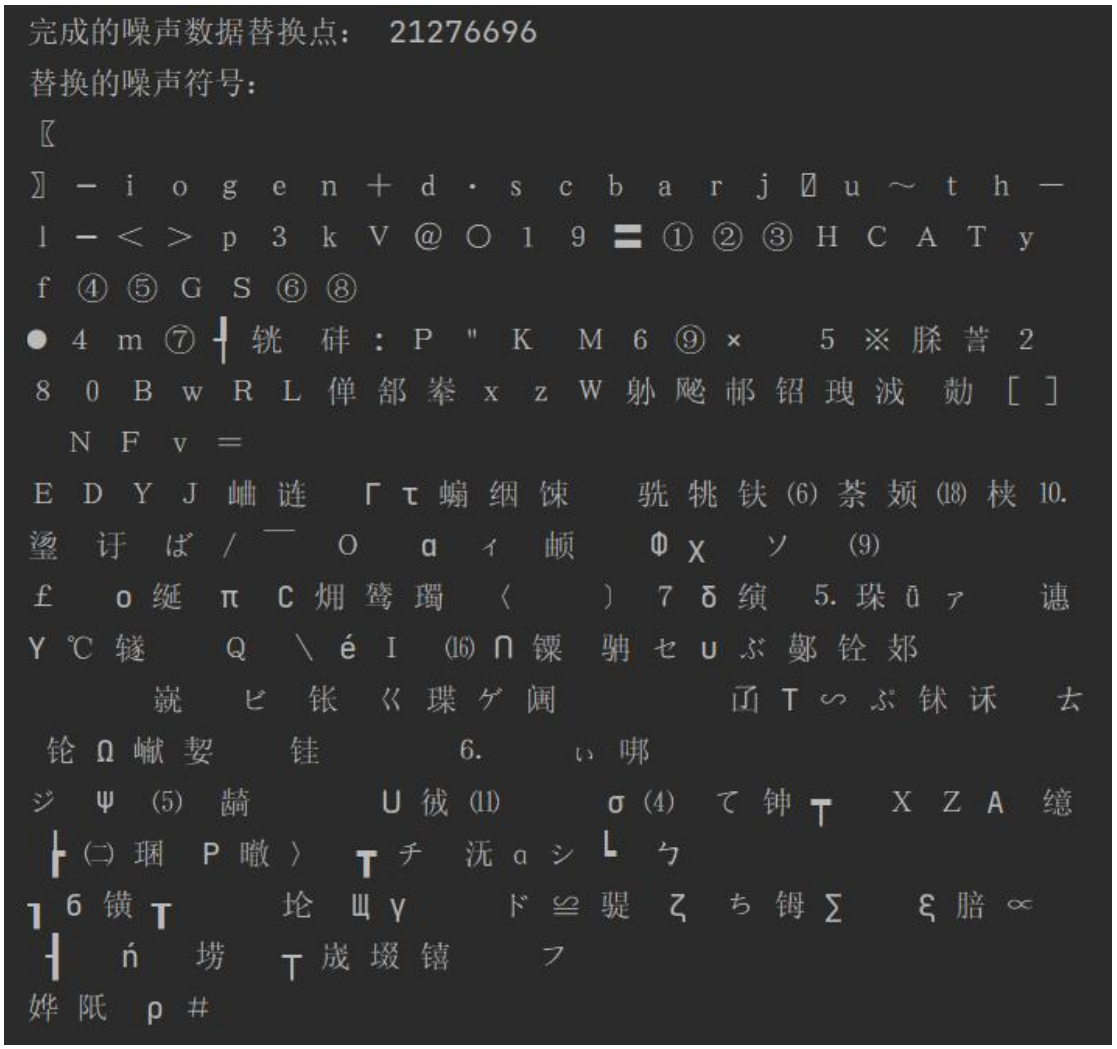
$$\begin{aligned} &P(S) = P(w1, w2, w3, \ldots, wn) \\ &= P(W1) P(W2 |\ W1) P(W3 |\ W1, W2), \ldots P(Wn |\ W1, W2, \ldots, \quad W_{n-1}) \\ &\approx P(W1) P(W2 |\ W1) P(W3 |\ W2, W1), \ldots P(Wn |\ Wn-1, Wn-2) \end{aligned}$$

以此类推，可以让 k=2，叫做 trigrams，4-grams，5-grams，当 k=N-1，模型成为 n 元模型，即 N-grams。

四、实验结果及分析

本实验分别采用 k=1,2,3 时的 N-Gram 模型进行对比分析，实验结果如下：

噪声数据替换：



1-Gram 结果：

```
词库总词数： 4314424  不同词的个数： 173004
出现频率前10的1-gram词语： [('的', 115604), ('了', 104527),
('他', 64712), ('是', 64457), ('道', 58623), ('我',
57483), ('你', 56679), ('在', 43691), ('也', 32606), ('这
', 32199)]
entropy_1gram: 12.1667882280937
```

2-Grams 结果：

```
词库总词数： 4255108    不同词的个数： 1950584
出现频率前10的2-gram词语：
道你 5738
叫道 5009
道我 4953
笑道 4271
听得 4202
都是 3905
了他 3638
他的 3497
也是 3201
的一声 3102
entropy_2gram: 6.9449780657055555
```

3-Grams 结果：

```
辚     Q 词库总词数： 4196041    不同词的个数： 3482118
出现频率前10的3-gram词语：
只听得 1611
忽听得 1137
站起身来 733
哼了一声 573
笑道你 566
. . .  549
吃了一惊 534
    513
点了点头 503
啊的一声 481
entropy_3gram: 2.3034876915483387
```

通过结果分析，当 k 值取得越大，即考虑前后词语关联的程度越大时，不同词出现个数增加，因而信息熵越小，因为分词之后得到文本的词组分布变得越简单，使得固定的词数量越多，固定的词能减少由字或者短词打乱文章的机会，使得文章变得更加有序，减少了由字组成词和组成句子的不确定性，因而降低了文本所含信息的信息量，也就是信息熵的大小。