成 绩 _____

北京航空航天大學

BEIHANG UNIVERSITY

深度学习与自然语言处理

第一次作业

院（系）名称	自动化科学与电气工程学院
专 业 名 称	电子信息
学 生 学 号	ZY2103202
学 生 姓 名	黄君辉
指 导 教 师	秦曾昌

2022 年 4 月

一　　中文信息熵的计算

一、问题描述

参考文章 Peter Brown 的文章：An Estimate of an Upper Bound for the Entropy of English，分别以词和字为单位计算中文的金庸小说全集的平均信息熵。

二、方法介绍

2.1 信息熵

信息，泛指人类社会传播的一切内容，在我们生活中随处可见，但却又十分抽象。如何具体地，量化地对信息进行描述，就需要用到“信息熵”的概念。1948 年，香农指出信息是用来消除随机不确定性的东西，并借用了热力学中熵的概念提出了“信息熵”的概念，来解决信息的度量问题。信息熵具有以下三个性质：

1. 单调性，发生概率越高的事件，其携带的信息量越低；
2. 非负性，信息熵可以看作为一种广度量，非负性是一种合理的必然；
3. 累加性，即多随机事件同时发生存在的总不确定性的量度是可以表示为各事件不确定性的量度的和，这也是广度量的一种体现。

对于任意一个离散随机变量X=$\{x_1, x_2, \cdots\}$的概率分布为$P(X = x_k) = p_k, k = 1,2, \cdots$，它的信息熵定义如下，信息熵的单位为比特(bit)。

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

把各种可能表示出的信息量乘以其发生的概率之后求和，就表示了整个系统所有信息量的一种期望值。变量的不确定性越大，信息熵越大，即确定变量需要的信息量就越大。从这个角度来说信息熵还可以作为一个系统复杂程度的度量，如果系统越复杂，出现不同情况的种类越多，那么他的信息熵是比较大的。对上式进行推广，对两个离散随机变量 X 和 Y，定义他们的联合熵 (Joint Entropy) 为：

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

联合熵表征了两事件同时发生系统的不确定度。条件熵 (Conditional Entropy) $H(X|Y)$则表示在已知随机变量Y的条件下随机变量X的不确定性，计算公式如下。

$$H(X|Y)=\sum_{y\in Y}p(y)H(X|Y=y)=-\sum_{y\in Y}p(y)\sum_{x\in X}p(x|y)logp(x|y)$$

当使用另一个变量Y对原变量X进行分类后，原变量X的信息量增加，不确定性减小，因此计算得到的信息熵也会减小。在自然语言处理中，信息熵只反映内容的随机性（不确定性）和编码情况，与内容本身无关。

2.2 统计语言模型

假定S表示某一个有意义的句子，由一连串特定顺序排列的词$\{w_1,w_2,\cdots w_n\}$组成，n为句子的长度。由于S是由这些词按照特定顺序排列而成，因此S在文本中出现的可能性$P(S)=P(w_1,w_2,\cdots w_n)$，利用条件概率公式可得

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

其中$P(w_1)$表示第一个词w_1出现的概率；$P(w_2|w_1)$是在已知第一个词的前提下，第二个词出现的概率；以此类推。当句子长度过长时，$P(w_n|w_1,w_2,\ldots,w_{n-1})$的可能性太多，根据马尔可夫假设：任意一个词$w_i$出现的概率只同它前面的$n-1$词有关，称为$N$元模型。当 N=1 时，每个词出现的概率与其他词无关，为一元模型，对应 S 的概率变为：

$$P(S)=P(w_1)P(w_2)P(w_3)\ldots P(w_i)\ldots P(w_n)$$

当 N=2 时，每个词出现的概率与其前一个词相关，为二元模型，对应 S 的概率变为：

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

三、思路说明及结果

3.1 思路说明

问题中需要处理的数据为中文的 16 本金庸小说，数据格式为 txt 文本格式，由于原始文本中存在广告、无用或重复的中英文特殊符号，为了获得更为准确的信息熵计算结果，需要对数据进行预处理。

以字为单位计算平均信息熵，直接读取需要计算的 txt 文件，去除掉文中多余的广告等无用信息后，统计每个字出现的次数，得到词频表，进而就可以计算信息熵。而以词为单位计算平均信息熵时，需要根据上下文关系将一个汉字序列切分成一个一个单独

的词。为了方便处理，可以利用 jieba 进行分词，得到所需要的 txt 格式语料库。jieba 是 python 中的一个中文分词库，是利用统计机器学习模型学习词语切分的规律（称为训练），从而实现对未知文本的切分。jieba 支持三种分词模式：

- 精确模式：将句子最精确的分开，适合文本分析
- 全模式：句子中所有可以成词的词语都扫描出来，速度快，不能解决歧义
- 搜索引擎模式：在精确的基础上，对长词再次切分，提高召回

本次作业中采用精确模式将文本进行划分，并使用一元和二元模型对每本书按字划分和按词划分的情况分别进行信息熵计算。对于二元模型，需要考虑上下文关系，不能直接去掉所有标点符号得到无分隔的语料，应该通过中文停词表清理语料，生成由停词分行的 txt 格式语料库，并统计每个二元词组在语料库中出现的频数，得到二元模型词频表，进而计算信息熵。

3.2 程序说明

文件读取及预处理部分代码如下所示：

```
def getCorpus(self, rootDir):
    corpus = []
    Character= u'[a-zA-Z0-9’!"#$%&\'()*+,-./:：;<=>?，。?、…【】《》？“”‘’！[\\]^_`{|}~]+'
# 去除文字外所有字符
    ad = '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'#去除广告
    count=0
    with open(rootDir, "r", encoding='ansi') as file:
        filecontext = file.read();
        filecontext = re.sub(r1, '', filecontext)
        filecontext = filecontext.replace("\n", '')
        filecontext= filecontext.replace(" ", '')
        filecontext = filecontext.replace(ad, '')
        #seg_list=jieba.cut(filecontext,cut_all =True)#按字计算时不需要进行分词
        #corpus += seg_list
```

```
            count += len(filecontext)
            corpus.append(filecontext)
        return corpus, count
```

词频统计部分代码如图所示：

```
#一元词频统计
def get_unigram_tf(tf_dic, words):
        for i in range(len(words)-1):
            tf_dic[words[i]] = tf_dic.get(words[i], 0) + 1
#  二元词频统计
def get_bigram_tf(tf_dic, words):
        for i in range(len(words)-1):
            tf_dic[(words[i], words[i+1])] = tf_dic.get((words[i], words[i+1]), 0) + 1
```

信息熵计算部分代码如下：

```
#一元模型
for uni_word in words_tf.items():
            entropy.append(-(uni_word[1]/words_len)*math.log(uni_word[1]/words_len, 2))
#二元模型
for bi_word in bigram_tf.items():
            jp_xy = bi_word[1] / bigram_len    #  计算联合概率 p(x,y)
            cp_xy = bi_word[1] / words_tf[bi_word[0][0]]    #  计算条件概率 p(x|y)
            entropy.append(-jp_xy * math.log(cp_xy, 2))    #  计算二元模型的信息熵
```

3.3 运行结果

按字划分时，16 本书信息熵的计算结果如下表所示：

名称	字数	一元模型信息熵（比特/字）	二元模型信息熵（比特/字）
白马啸西风	64061	8.59885	4.56253
碧血剑	420068	9.45091	5.85253
飞狐外传	378777	9.30421	5.73682
连城诀	199412	9.11518	5.34571

鹿鼎记	1047468	9.24284	5.95121
三十三剑客图	53682	9.67532	4.83131
射雕英雄传	794124	9.37655	6.03406
神雕侠侣	835666	9.34015	5.99407
书剑恩仇录	438338	9.46911	5.76879
天龙八部	1040707	9.36593	6.07479
侠客行	317099	9.10154	5.53564
笑傲江湖	833372	9.19615	5.87477
雪山飞狐	119513	9.1328	5.07404
倚天屠龙记	830439	9.36870	5.99919
鸳鸯刀	32552	8.75369	4.23183
越女剑	14803	8.55327	3.71197

按词划分时，16 本书信息熵的计算结果如下表所示：

名称	词数	一元模型信息熵（比特/字）	二元模型信息熵（比特/字）
白马啸西风	42207	9.49565	4.07855
碧血剑	246405	11.69014	4.99266
飞狐外传	224513	11.46932	4.98548
连城诀	122177	10.84085	4.66206
鹿鼎记	629266	11.25894	5.71454
三十三剑客图	31551	11.66358	2.95867
射雕英雄传	480018	11.55264	5.47727
神雕侠侣	502841	11.51878	5.46854
书剑恩仇录	255908	11.68502	5.01994
天龙八部	623475	11.57242	5.63362
侠客行	190545	10.99487	4.90923
笑傲江湖	490987	11.32648	5.59172
雪山飞狐	73383	10.77300	4.09864
倚天屠龙记	487540	11.63975	5.50211

鸳鸯刀	20658	9.71726	3.34471
越女剑	9293	9.42809	2.78805

3.4 结果分析

可以看出，按字计算和按词计算得到的信息熵有较大区别。

1) 就语料库而言，16 本书字数差距较大，最多的《鹿鼎记》有 104w 字，而最少的《越女剑》仅有 1.4w 字，词数更是只有 9k 左右。由于本次作业分词采用的是 jieba 库的精准模式，所以词数总体来说小于字数
2) 就一元模型而言，按字计算的方式得到每本书的信息熵差距不大，可以看出，字数不同的书信息熵相差却不是特别大；说明信息熵只是表示信息的混乱程度，而与其大小无直接关系。并且，按字计算得到的信息熵基本上均小于按词计算得到的信息熵，这说明词相较于字可能有更多的排列组合方式，不确定性更高，因此信息熵也越大。
3) 就二元模型而言，其信息熵明显小于一元模型得到的信息熵，因此证明了随着关联信息的增加，信息的整体不确定度下降，信息熵减小。但是就不同的计算方式而言，按字计算得到的信息熵反而大于按词计算的信息熵，这说明，两字组成的词比两个词组成的词组有着更大的不确定性。

四、收获、体会及建议

通过此次作业，我对信息熵的定义和计算方式有了更深刻的理解，学习了 jieba 库的基本使用方法，对自然语言处理的相关内容有了基础性的认识。

五、参考文献及代码链接

信息熵到底是什么_saltriver 的博客-CSDN 博客_信息熵

信息熵及其相关概念_Northan 的博客-CSDN 博客_信息熵

深度学习与自然语言处理实验——中文信息熵的计算_DiliDili_Q 的博客-CSDN 博客

Jinhan-Lin/DL-NLPproject1: first homework of DL-NLP course (github.com)