中文平均信息熵的计算（分别以字和词为单位）

彭海琅 ZY2203322

plunck@buaa.edu.cn

摘要

信息熵（information entropy）是信息论的基本概念，描述信息源各可能事件发生的不确定性，发生概率越高的事件，其所携带的信息熵越低。信息熵的提出解决了对信息的量化度量问题。中文被视为世界主流语言中信息熵最高的，一般认为单个英文字母的信息熵为 3.9bits，中文平均单个汉字信息熵是 9.56bits。本文基于马尔科夫链和大数定理，使用 Python 语言计算中文的平均信息熵，并对分别以字和词为基本单位的情况进行对比分析。

导论

一、信息熵

信息论之父 C. E. Shannon 在 1948 年发表的论文“通信的数学理论(A Mathematical Theory of Communication)“中指出，任何信息都存在冗余，冗余大小与信息中每个符号（数字、字母或单词）的出现概率或者说不确定性有关[1]。他提出了一个表征符号系统中单位符号平均信息量的指标——信息熵，还给出了一个计算信息熵的公式，这个公式十分简洁：

$$-\sum_{i=1}^{n} p_i log p_i$$

公式里的p_i指的是某种符号系统中，某个符号出现的频率。

信息熵具有三个基本性质：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。

根据参考论文[2]，对于文本$X = \{\dots, X_{-2}, X_{-1}, X_0, X_1, X_2, \dots\}$，它的信息熵定义为

$$H(X) \equiv H(P) \equiv -E_P log P(X_0 | X_{-1}, X_{-2}, \dots)$$

二、N-Gram 语言模型

N-Gram 是一种基于统计语言模型的算法。它的基本思想是将文本里面的内容按照字节进行大小为 N 的滑动窗口操作，形成了长度是 N 的字节片段序列。它基于这样一种假设，第 N 个词的出现只与前面 N-1 个词相关，而与其它任何词都不相关，整句的概率就是各个词出现概率的乘积。这些概率可以通过直接从语料中统计 N 个词同时出现的次数得到。常用的是二元的 Bi-Gram 和三元的 Tri-Gram。

如果我们有一个由 m 个词组成的序列（或者说一个句子），我们希望算得概率$p(w_1, w_2, \dots, w_m)$，根据链式规则，可得

$$P(w_1, w_2, \dots, w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2) \dots P(w_m|w_1, \dots, w_{m-1})$$

这个概率显然并不好算，不妨利用马尔科夫链的假设，即当前这个词仅仅跟前面几个有限的 n-1 个词相关，因此也就不必追溯到最开始的那个词，这样便可以大幅缩减上述算式的长度。即

$$P(w_1, w_2, \dots, w_m) = P(w_{\mathrm{m}}|w_{\mathrm{m-n+1}}, \dots, w_{\mathrm{m-1}}) * P(w_{\mathrm{m-n+1}}, \dots, w_{\mathrm{m-1}})$$

特别地，下面给出一元模型，二元模型，三元模型的定义。

当 n=1，一个一元模型(unigram model)即为

$$p(w_1, w_2, \dots, w_m) = \prod_{i=1}^{m} P(w_i)$$

当 n=2，一个二元模型（bigram model)即为 ：

$$p(w_1, w_2, \dots, w_m) = \prod_{i=1}^{m} P(w_i|w_{i-1})$$

当 n=3，一个三元模型（trigram model)即为 ：

$$p(w_1, w_2, \dots, w_m) = \prod_{i=1}^{m} P(w_i|w_{i-2}w_{i-1})$$

则有一元模型的信息熵计算公式为：

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

其中$P(x)$可近似于每个词在语料库中的出现频率。

二元模型的信息熵计算公式为：

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

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率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：

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

其中联合概率$P(x,y,z)$可近似等于每个三元词组在语料库中出现的频率，条件概率$P(x|y,z)$可近似等于每个三元词组在语料库中出现的频率与以该三元词组的前两个词为词首的三元词组的频数的比值。

实验内容

一、中文语料预处理

读取停词表，剔除语料中无意义的乱码，删除换行符及分页符等无用的符号，删除停词表中的所有符号和词汇，最后得到一个纯文本长字符串。

```
def read_data(path):
    with open(path, 'r', encoding='ANSI') as f:
        data = f.read()
        f.close()

    delete = ['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', '新语丝电子文库',
      '\u3000', '\n', '。', '？', '！', '，', '、', '；', '：', '、',
      '『', '』', ' (', ') ', '…', '「', '」', '\ue41b', '<', '>', '+', '\x1a', '\ue42b']

    with open('./cn_stopwords.txt', 'r', encoding='utf-8') as f:
        for a in f:
            if a != '\n':
                delete.append(a.strip())

    for a in delete:
        data = data.replace(a, '')

    return data
```

二、计算一元、二元、三元的词频

使用 Python 中的字典结构存储一元组、二元组、三元组出现的频次。

```
def get_unigram_wf(words):
    dic = {}
    for i in range(len(words)):
        dic[words[i]] = dic.get(words[i], 0) + 1
    return dic
```

三、计算一元、二元、三元组的信息熵

依据导论中的公式计算信息熵并保存输出。

```
def unigram_model(dic):
    begin = time.time()
    unigram_sum = sum([item[1] for item in dic.items()])  # 一元模型分词个数
    entropy = 0  # 一元模型信息熵
    for item in dic.items():
        entropy += -(item[1] / unigram_sum) * math.log(item[1] / unigram_sum, 2)
    entropy = round(entropy, 4)
    end = time.time()
    runtime = round(end - begin, 4)
    return entropy,runtime
```

实验结果

一、词频分析

从词频统计可以发现，人物名字出现的频率较高，以《倚天屠龙记》为例，按字划分时，出现频率最高的一元、二元、三元组是：

```
频率最高的一元字：('道', 11678)
频率最高的一元字：('张', 7493)
频率最高的一元字：('中', 6095)
频率最高的一元字：('说', 5328)
频率最高的一元字：('忌', 5174)
频率最高的二元字：(('张', '忌'), 4667)
频率最高的二元字：(('说', '道'), 1692)
频率最高的二元字：(('赵', '敏'), 1248)
频率最高的二元字：(('谢', '逊'), 1209)
频率最高的二元字：(('翠', '山'), 1167)
频率最高的三元字：((('张', '翠'), '山'), 1146)
频率最高的三元字：((('张', '忌'), '道'), 652)
频率最高的三元字：((('殷', '素'), '素'), 554)
频率最高的三元字：((('张', '三'), '丰'), 450)
频率最高的三元字：((('灭', '绝'), '师'), 438)
```

按词划分时，出现频率最高的一元、二元、三元组是：

```
频率最高的一元词：('道', 4736)
频率最高的一元词：('张忌', 3331)
频率最高的一元词：('便', 2717)
频率最高的一元词：('说', 2047)
频率最高的一元词：('说道', 1648)
频率最高的二元词：(('张忌', '道'), 467)
频率最高的二元词：(('杨', '逍'), 401)
频率最高的二元词：(('笑', '道'), 320)
频率最高的二元词：(('少林', '派'), 212)
频率最高的二元词：(('突然', '间'), 177)
频率最高的三元词：((('杨', '逍'), '道'), 44)
频率最高的三元词：((('乾坤', '挪移'), '心法'), 39)
频率最高的三元词：((('张忌', '笑'), '道'), 36)
频率最高的三元词：((('杨', '逍'), '韦笑'), 32)
频率最高的三元词：((('金庸', '倚天'), '屠龙记'), 30)
```

从中可以看出人们基于一定的语料库，可以利用 N-Gram 来预计或者评估一个句子是否合理，N-Gram 语言模型具备一定合理性。

二、按字分割的 N-Gram 模型信息熵

单位/bit	一元	二元	三元
白马啸西风	9.2288	4.0789	1.2090
碧血剑	9.7609	5.6684	1.7886
飞狐外传	9.6337	5.5639	1.8608
连城诀	9.5195	5.0837	1.6361
鹿鼎记	9.6641	6.0154	2.402
三十三剑客图	10.0118	4.2776	0.6491
射雕英雄传	9.7554	5.9622	2.1915
神雕侠侣	9.5595	6.0525	2.3701
书剑恩仇录	9.7627	5.5921	1.8571
天龙八部	9.7879	6.1104	2.3454
侠客行	9.4393	5.3722	1.8144
笑傲江湖	9.5200	5.8509	2.3561
雪山飞狐	9.5055	4.7943	1.2989
倚天屠龙记	9.7097	5.9787	2.2723
鸳鸯刀	9.2115	3.6510	0.8951
越女剑	8.7854	3.1039	0.8430
全文	9.9534	7.0319	3.4910

随着 N 的取值增大，文本的信息熵不断减小，说明上下文关系使得文本的不确定性下降了，这与我们的常识相符合。

三、按词分割的 N-Gram 模型信息熵

单位/bit	一元	二元	三元
白马啸西风	11.1917	2.6993	0.2658
碧血剑	12.9338	3.733	0.3833
飞狐外传	12.7401	3.7633	0.3811
连城诀	12.2949	3.315	0.3005
鹿鼎记	12.9095	4.6818	0.6597
三十三剑客图	12.4442	1.657	0.0693
射雕英雄传	13.1387	4.3283	0.4665
神雕侠侣	12.476	4.6953	0.7185
书剑恩仇录	12.7835	3.9215	0.4311
天龙八部	13.235	4.5047	0.558
侠客行	12.3915	3.7073	0.4466
笑傲江湖	12.63	4.567	0.7274
雪山飞狐	12.149	2.7641	0.2293
倚天屠龙记	13.0215	4.4137	0.5641
鸳鸯刀	10.9822	2.1029	0.1884
越女剑	10.2625	1.7188	0.231
全文	13.8942	6.1768	1.0001

按词分割时，一元模型的信息熵大于按字分割，这可能是因为词的种类比单字多，使得可能性增加了，从而增加了不确定性；到了二元和三元模型中，词的组合数比字的组合数要少了，所以信息熵开始低于按字分割。

结论

本文基于马尔科夫链和大数定理，使用 Python 语言计算中文的平均信息熵，并对分别以字和词为基本单位的情况进行对比分析，实验得出，在 N-Gram 语言模型中，随着 N 的增大，上文关系的引入使得文本的不确定性下降，文本的信息熵降低，这与我们的常识相符。对中文文本分别按字分割与按词分割时，按词分割 1-Gram 模型的信息熵更大，按字分割 2-Gram 和 3-Gram 模型的信息熵更大，这可能是因为词的种类数比字多，但是词的组合数比字的组合数要少，这种可能性的差异造成了信息熵的差异。

参考资料

[1] Shannon, C. E . A Mathematical Theory of Communication[J]. Bell Systems Technical Journal, 1948, 27(4):623-656.
[2] Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai. 1992. An estimate of an upper bound for the entropy of English. Comput. Linguist. 18, 1 (March 1992), 31–40.