计算中文文本信息熵

杨洪易
yhongyi@buaa.edu.cn

摘要:

本文以给定的中文小说作为文本库，通过去除标点与停词生成语料库，并利用 jieba 对小说文本进行分词，分别对各个文本的语料库内容做了一元、二元与三元模型的信息熵计算。

简介:

一.信息熵

熵，泛指某些物质系统状态的一种量度，某些物质系统状态可能出现的程度。亦被社会科学用以借喻人类社会某些状态的程度。熵的概念是由德国物理学家克劳修斯于 1865 年所提出。最初是用来描述“能量退化”的物质状态参数之一，在热力学中有广泛的应用。熵的本质是一个系统“内在的混乱程度”。它在控制论、概率论、数论、天体物理、生命科学等领域都有重要应用，在不同的学科中也有引申出的更为具体的定义，按照数理思维从本质上说，这些具体的引申定义都是相互统一的，熵在这些领域都是十分重要的参量。

信息熵的定义公式:

$$H(x) = -\sum p(x)\log p(x)$$

并且规定：0 log（0） = 0。

信息熵的三个性质:

信息论之父克劳德·香农给出的信息熵的三个性质:

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。

二.jieba 分词

jieba 分词主要是基于统计词典，构造一个前缀词典；然后利用前缀词典对输入句子进行切分，得到所有的切分可能，根据切分位置，构造一个有向无环图（DAG）；通过动态规划算法，计算得到最大概率路径，也就得到了最终的切分形式。

特点:

1.支持三种分词模式：精确模式，全模式，搜索引擎模式

2.支持繁体分词

3.支持自定义词典

4.MIT 授权协议

涉及算法:

1.基于前缀词典实现词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG），采用动态规划查找最大概率路径，找出基于词频的最大切分组合;

2.对于未登录词，采用了基于汉字成词能力的 HMM 模型，采用 Viterbi 算法进行计算；
3.基于 Viterbi 算法的词性标注；
4.分别基于 TFIDF 和 TextRank 模型抽取关键词；

模型：

语言模型就是用来计算一个句子的概率的模型，也就是判断一句话是否是人话的概率。给定一个句子（词语序列）：

$$S = W_1, W_2, ..., W_k$$

它的概率可以表示为：

$$P(S) = P(W_1, W_2, ..., W_k) = P(W_1)P(W_2 \mid W_1)...P(W_k \mid W_1, W_2, ..., W_k)$$

因此有如下信息熵模型：

一元模型：

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

其中 p(x)为每个词在语料库中出现的频率，可以看作是单个词所携带的信息熵。

N 元模型：

将自然语言句子视作 N-1 阶马尔可夫模型，即规定句子中某词出现的概率只同它前面出现的 N-1 个词有关。常见的二元模型：

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

程序实验：

1.环境
Anaconda3 python3.9.16
2. 文本库、标点表、停词表
文本库为中文小说，标点与停词为“cn_punctuation.txt”与“cn_stopwords.txt”，按照要求对文本库内所有标点与停词进行了去除，生成“novel_sentence.txt”文件，每行为一短句。
随后，在处理过程中读取“novel_sentence.txt”并用 jieba 分词。

```
#读标点
def read_punctuation_list(path):
    punctuation = [line.strip() for line in open(path, encoding='UTF-8').readlines()]
    punctuation.extend(['\n', '\u3000', '\u0020', '\u00A0'])
    return punctuation
#读停词
def read_stopwords_list(path):
    stopwords = [line.strip() for line in open(path, encoding='UTF-8').readlines()]
    return stopwords
```

```
#按行读取 并且分词
for line in f:
    if line != '\n':
        corpus.append(line.strip())
        count += len(line.strip())
for line in corpus:
    for x in jieba.cut(line):
        split_words.append(x)
        words_len += 1
```

3.数据处理

get_tf 与 get_bigram_tf 分别获取一元词频与二元词频，get_bi_tf 与 get_trigram_tf 获取非句尾的二元词频与三元词频，按公式计算信息熵。

```
#一元模型获取词频率
def get_tf(words):
    tf_dic = {}
    for i in range(len(words)-1):
        tf_dic[words[i]] = tf_dic.get(words[i], 0) + 1
        # print(tf_dic.get(w,0))
    return tf_dic

#二元模型获取词频率
def get_bigram_tf(words):
    tf_dic = {}
    for i in range(len(words)-1):
        tf_dic[(words[i], words[i+1])] = tf_dic.get((words[i], words[i+1]), 0) + 1
        # print(tf_dic.get(w,0))
    return tf_dic

#三元元模型获取词频率
#非句子末尾二元词频统计
def get_bi_tf(words):
    tf_dic = {}
    for i in range(len(words)-2):
        tf_dic[(words[i], words[i+1])] = tf_dic.get((words[i], words[i+1]), 0) + 1
    return tf_dic
# 三元模型词频统计
def get_trigram_tf( words):
    tf_dic = {}
    for i in range(len(words)-2):
        tf_dic[((words[i], words[i+1]), words[i+2])] = tf_dic.get(((words[i], words[i+1]), words[i+2]), 0) + 1
    return tf_dic
```

4.实验结果

分别以字与词为单位的一至三元信息熵如下表所示：

（词）	一元信息熵	二元信息熵	三元信息熵
三十三剑客图	11.827	2.428	0.12
书剑恩仇录	11.839	4.8	0.666
侠客行	11.365	4.658	0.711
倚天屠龙记	11.864	5.276	0.964
天龙八部	12.043	5.51	0.896
射雕英雄传	12.081	5.278	0.75
白马啸西风	10.55	3.399	0.414
碧血剑	11.969	4.662	0.598
神雕侠侣	11.773	5.316	0.904
笑傲江湖	11.541	5.448	1.087
越女剑	9.957	2.164	0.278
连城诀	11.357	4.238	0.508
雪山飞狐	11.248	3.668	0.444

飞狐外传	11.723	4.698	0.645
鸳鸯刀	10.541	2.65	0.275
鹿鼎记	11.703	5.491	1.159

（字）	一元信息熵	二元信息熵	三元信息熵
三十三剑客图	10.007	4.284	0.651
书剑恩仇录	9.758	5.598	1.862
侠客行	9.435	5.38	1.82
倚天屠龙记	9.706	5.983	2.276
天龙八部	9.782	6.115	2.352
射雕英雄传	9.752	5.965	2.196
白马啸西风	9.225	4.089	1.212
碧血剑	9.755	5.675	1.796
神雕侠侣	9.663	6.002	2.283
笑傲江湖	9.516	5.856	2.361
越女剑	8.781	3.109	0.842
连城诀	9.515	5.09	1.639
雪山飞狐	9.501	4.801	1.303
飞狐外传	9.63	5.569	1.865
鸳鸯刀	9.21	3.656	0.896
鹿鼎记	9.658	6.02	2.41

结论:

1.横向对比：比较一至三元信息熵。元数越多，信息熵越小。这可能是因为随元数增加词与字的排列更加固定，使得信息熵降低。
2.纵向对比：比较字与词的信息熵。在一元模型下词比字的信息熵更大，随着元数增加字比词的信息熵更大。这可能是因为在一元模型下，字之间存在多种组词方式，使得词语信息熵较大：随着元数增加，仍有多种组词方式，但词间关系更加固定，相较之下词语信息熵较小。

参考资料:

[1]深度学习与自然语言处理：中文信息熵的计算_中文 信息熵 计算_qq_40412713 的博客-CSDN 博客
[2] NLP——语料库信息提取和处理方法_语料库的收集与整理_季建豪的博客-CSDN 博客