中文信息熵计算

吴金旺 ZY2203810

wjwpoi@buaa.edu.cn

实验原理

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。20 世纪 40 年代，香农（C.E.Shannon）借鉴了热力学的概念，把信息中排除了冗余后的平均信息量称为“信息熵”，并给出了计算信息熵的数学表达式。信息熵的提出解决了对信息的量化度量问题。

信息熵的数学公式为：

$$H(x) = E\big[-\log\big(P(x)\big)\big] = -\sum_{x\in X} P(x)\log\big(P(x)\big) \tag{1}$$

式（1）中的对数通常以 2 为底，单位为比特。

对于任何一个事件，通常来说，它的不确定性越大，那么其信息熵也越大。反之，如果不确定性越小，那么信息熵也越小。越是不确定的事件，如果我们得到了一条有关的信息，那么信息量就会很大，即信息熵与信息的价值成正相关。反之，基本确定的事件，信息的价值也较小，信息熵也相对较小。

语言模型（Language Model, LM）是对自然语言的建模，从机器学习的角度来看，也可以认为语言模型是对语句的概率分布的建模。简单来说，人的语言是有上下文关系的，所说的后文必然与前文有很大联系。语言模型利用这一点，学习人的语言能力。

给定一个句子序列$S = w_0 w_1 \dots w_n$，我们首先想要知道它是否符合语言的规则，因此我们用一个联合概率$P(w_0, w_1, \dots, w_n)$来衡量它符合规则的概率，根据链式法则，这个概率又可表示为：

$$\begin{aligned} P(S) = P(w_0, w_1, \dots, w_n) &= P(w_0)P(w_1|w_0)P(w_2|w_0 w_1)\dots P(w_n|w_0 w_1 \dots w_{n-1}) \\ &= \prod_i P(w_i|w_0 w_1 \dots w_{i-1}) \end{aligned} \tag{2}$$

因此，语言模型实际上就是一系列条件概率的连乘，然而，当句子序列的长

度过长，即n过大的时候，很难计算这么一长串的条件概率。俄国数学家马尔可夫假设任意一个词w_i出现的概率只同它前面的词w_{i-1}有关，这种假设称为马尔可夫假设，此时有：

$$\begin{aligned} P_2(S) &= P(w_0)P(w_1|w_0)P(w_2|w_1)\dots P(w_n|w_{n-1}) \\ &= \prod_i P(w_i|w_{i-1}) \end{aligned} \tag{3}$$

这种模型也被称为二元（2-gram）模型，类似的，有一元模型，三元模型和任意元模型。

其中一元（1-gram）模型认为每个词只与它本身有关，可表示为：

$$\begin{aligned} P_1(S) &= P(w_0)P(w_1)P(w_2)\dots P(w_n) \\ &= \prod_i P(w_i) \end{aligned} \tag{4}$$

三元(3-gram)模型认为每个词与它前面两个词有关，可表示为：

$$\begin{aligned} P_3(S) &= P(w_0)P(w_1|w_0)P(w_2|w_0w_1)P(w_3|w_1w_2)\dots P(w_n|w_{n-2}w_{n-1}) \\ &= \prod_i P(w_i|w_{i-2}w_{i-1}) \end{aligned} \tag{5}$$

通常，一元模型使用较少，因为它完全忽略了词之间的上下文关系，二元和三元乃至更高元的模型则使用的较多。

语言模型的好处是在训练时不需要任何标注。如今在 NLP 领域大火的 GPT 系列模型也是一种类似语言模型，仅仅通过大量的未标注文本数据就能充分的学习到人类的语言习惯乃至各种知识。

根据式（3-5），我们可以分别用一元、二元、三元语言模型对给定的中文文本进行建模，并计算信息熵，类似公式（1），三种语言模型的信息熵分别被定义为：

$$H_1(w) = -\sum_i P(w_i)\log\big(P(w_i)\big) \tag{6}$$

$$H_2(w) = -\sum_i P(w_{i-1}w_i)\log\big(P(w_i|w_{i-1})\big) \tag{7}$$

$$H_3(w) = -\sum_i P(w_{i-2}w_{i-1}w_i)\log\big(P(w_i|w_{i-2}w_{i-1})\big) \tag{8}$$

所有的概率都可以用频率来近似表示，以三元模型为例，有：

$$P(w_{i-2}w_{i-1}w_i) = \frac{N(w_{i-2}w_{i-1}w_i)}{N(w)-2} \tag{9}$$

$$P(w_i|w_{i-2}w_{i-1}) = \frac{N(w_{i-2}w_{i-1}w_i)}{N(w_{i-2}w_{i-1})} \tag{10}$$

N代表括号内组合出现的数目，其中$N(w)$代表总词数，为了保证统一性，从第三个词开始计算条件概率，因此总共的三元组的数目等于总次数-2。因此，计算信息熵实际上是一个统计问题。

实验内容

首先，在正式统计文本之前，需要对数据进行预处理。

文章包含了一些换行符，还有一小部分英文字母，这些都是需要删除的。在一些最先进的语言模型中，标点符号也会考虑在其中，但是在这里，为了方便分词和处理，将删去所有的标点符号，只留下纯文本的问题。我使用正则匹配来进行文本的过滤。

然后，在按照词统计信息熵之前，需要先分词，因此使用了结巴分词来实现，同时分词后，还对停用词（stop words）进行了删除。对于按照字来统计信息熵的情况，没有做任何的删除处理。

统计词频较为简单，笔者使用了 python 中的字典（dictionary）来存储所有的字（词）组合，通过滑动窗口，不断统计每种组合出现的频率。

一元模型只需统计每个字（词）的频率即可计算；二元模型还需统计相邻两个字（词）的出现频率；三元模型则额外加上了相邻三个字（词）的出现频率。值得注意的是，统计时，每个组合的顺序也要考虑，顺序不同视为不同的组合。

根据式（6-8），利用 2.2 节统计的词频，我们可以轻易计算出每篇小说在一元/二元/三元模型设定下的信息熵。

对于整个语料库的信息熵，我们将每篇小说中的各种字（词）组合出现数目相加，然后用相加后的结果计算整体的信息熵。

实验结果

按照 2 中所述的实验方法，计算信息熵，得到结果如下

表 1 按照字计算的信息熵

书名	一元模型	二元模型	三元模型
白马啸西风	8.91505	4.58302	1.61874
碧血剑	9.45809	5.86518	2.34951
飞狐外传	9.31002	5.75319	2.40611
连城诀	9.17294	5.39795	2.15652
鹿鼎记	9.29069	5.98642	2.94918
三十三剑客图	9.67252	4.83403	0.98231
射雕英雄传	9.44170	6.06369	2.75902
神雕侠侣	9.37751	6.06339	2.84361
书剑恩仇录	9.46239	5.78224	2.39286
天龙八部	9.40501	6.12531	2.94904
侠客行	9.15375	5.59126	2.36860
笑傲江湖	9.20805	5.89754	2.87004
雪山飞狐	9.20760	5.15709	1.75614
倚天屠龙记	9.39518	6.02070	2.80438
鸳鸯刀	9.04371	4.21468	1.11261
越女剑	8.83132	3.64049	0.90785
全文	9.53302	6.72638	3.94882

信息熵的单位都是比特/词。显然，按照字计算中文的信息熵时，一元模型的信息熵最大，猜测是因为一元模型完全忽略了上下文，没有什么规律，因此不确定性很大。

二元和三元模型的信息熵都有所下降，这说明上下文关系可能确定了一部分字的出现，这使得确定性有所上升，且考虑的上下文越长，这种确定性越高。

值得注意的是，将所有语料的字频考虑在一起，计算总的信息熵时，结果发生了一定的上升。推测是由于不同文章的行文不同，用语不同，特别是一些名词不同，这导致了词频的大幅度变化，不确定性上升。

表 2 按照词计算的信息熵

书名	一元模型	二元模型	三元模型
白马啸西风	11.16109	2.89572	0.34832
碧血剑	12.91944	3.93553	0.41835
飞狐外传	12.66092	4.01577	0.44768
连城诀	12.24743	3.55734	0.35334
鹿鼎记	12.67209	4.96935	0.81978
三十三剑客图	12.57839	1.75646	0.08447
射雕英雄传	13.06910	4.56622	0.52538
神雕侠侣	12.87925	4.73241	0.62787
书剑恩仇录	12.75564	4.12108	0.48087
天龙八部	13.06935	4.80108	0.64555
侠客行	12.32764	3.96850	0.49527
笑傲江湖	12.56192	4.81887	0.77694
雪山飞狐	12.08518	3.04109	0.28264
倚天屠龙记	12.93542	4.65939	0.62568
鸳鸯刀	11.12239	2.13795	0.23268
越女剑	10.48776	1.72091	0.23292
全文	13.64171	6.50336	1.15033

按照词统计信息熵时，出现了与上一节类似的结果，同样是三元模型信息熵最低，一元模型信息熵最高。

但值得注意的是，按照词统计时，一元模型的信息熵比按字统计的信息熵更大；二元模型时信息熵基本相等，不过单看每一篇小说，按词统计的信息熵更小了；到了三元模型时，按词统计的信息熵则各种意义上都更小了。

就我个人理解而言，一元模型其实只与字（词）频有关，由于字可以自由组合成词，形成的词的总数更大，因此不确定性更高，信息熵也更高。二元模型下，由于词的组合出现了一定的“套路”，所以信息熵下降；三元模型时，这种“套路”更加明显，因此信息熵进一步下降。

实验总结

结合上述分析，我们发现，在利用三元模型并按照词统计的设定下，得到的中文信息熵最小，这说明我们在这种设定下能够把握一定的语言规律。可以预想的是，我们应该选择更高元的模型，即滑动窗口应该更大。同时，分词是必须的，对于中文，我们应该有更好的分词手段。

这次实验我们采用的语言模型虽然简单，也没有任何训练的内容，但它却是许多现阶段 SOTA 的模型的基础。实际上，如果我们将其中的概率由简单的计算频率，替换为用一个神经网络来拟合，那么实际上这样的模型就可以判断一句话是否符合人类语言习惯，乃至自动生成补全句子。所以，语言模型是十分有用的。