中文信息熵估计

李明昕

limx0204@buaa.edu.cn

摘要

本文以十六本金庸武侠小说作为语料库（corpus），训练了一个三元语言模型（trigram model）。并利用语言模型计算出交叉熵（cross entropy）作为上界，对中文的信息熵（entropy）进行了估计。在实验部分，分别以字、词为单位对语料库进行了切分，并分别将每本小说作为测试样例对信息熵进行估计，其中，分字模式下，对中文信息熵的估计为 5.4386；分词模式下，对中文信息熵的估计为 6.3195 [1]。

1 介绍

对于一个语言来说，信息熵的含义是文本中每个字符或符号的信息量大小，可以用来量化文本的信息复杂度和信息含量。本文使用了由金庸创作的十六本武侠小说作为语料，对中文的信息熵进行估计。当使用字为单位对这些语料进行切分时，共能切分出 963291 个单元（token），当以词为单位对这些语料进行切分时，共能切分出 614402 个单元。

本文的主要工作包括：

1. 对由十六本武侠小说构成的语料进行了清洗，并分别以字、词为单位对其进行了切分。用两个特殊单元（[BLK] 以及 [SEP]）对标点符号进行了替换；
2. 利用语料构建了由一元（unigram）、二元（bigram）以及三元（trigram）语言模型组合而成的语言模型；
3. 分别以每本武侠小说作为测试样例，利用语言模型计算得到的交叉熵对中文信息熵进行了估计。

2 方法

2.1 语料的预处理

在清洗语料库时，主要使用的方法是正则表达式替换，清洗的内容包括以下几个部分：

- 每本小说的开头和结尾都包括一段网站的广告，我在清洗的过程中删除了这部分广告；
- 语料中除了中文字符以外还包括其他字符，我在清洗的过程中只保留了中文字符以及常见的标点符号；

对于清洗过后的语料，我还将标点符号替换为了两个特殊的单元，具体来说：

- 对于逗号、省略号、顿号以及冒号，将他们替换为 [BLK]，表示文本之间的停顿；
- 对于句号、感叹号、问号以及分号，将他们替换为 [SEP]，表示两端文本之间的分割。

[1] 本文实验部分的代码见：https://github.com/8igfive/NLP2023/tree/main/HW1

2.2 语言模型的构建

语言可以看作是一个平稳随机过程（stationary stochastic process） [1]：

$$X = \{\ldots X_{-2}, X_{-1}, X_0, X_1, X_2, \ldots\}, \tag{1}$$

他的概率分布为 P ：

$$P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)\cdots P(X_n|X_{n-1}, \ldots, X_1). \tag{2}$$

但要直接构建这样的分布存在一定的困难。一种常见的简化方法是 n 元语言模型，也就是说每一个单元的概率只与其前 (n-1) 个单元有关，比如一元语言模型表示为：

$$P_1(X_1, X_2, \ldots, X_n) = P_1(X_1)P_1(X_2)\cdots P_1(X_n). \tag{3}$$

二元语言模型表示为：

$$P_2(X_1, X_2, \ldots, X_n) = P_2(X_1)P_2(X_2|X_1)\cdots P_2(X_n|X_{n-1}). \tag{4}$$

三元语言模型表示为：

$$P_3(X_1, X_2, \ldots, X_n) = P_3(X_1, X_2)P_3(X_3|X_2, X_1)\cdots P_3(X_n|X_{n-1}, X_{n-2}). \tag{5}$$

本文所要构建的语言模型参考了 [1]，具体来说是通过组合一元、二元以及三元语言模型来得到最终的语言模型：

$$M(X_3|X_2, X_1) = \lambda_3(X_2, X_1)P_3(X_3|X_2, X_1) + \lambda_2(X_2, X_1)P_2(X_3|X_2) + \lambda_1(X_2, X_1)P_1(X_3) + \lambda_0(X_2, X_1)P_0 \tag{6}$$

要得到该语言模型，需要将训练语料分为两部分，一部分占总训练预料的 90% ，用来统计 P_1, P_2, P_3。在得到 P_1, P_2, P_3 后，再用剩下的 10% 数据去训练 $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ 四个参数。具体来说，本文通过梯度下降算法，以 $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ 为可以训练的参数，通过训练使得语言模型 $M(X_3|X_2, X_1)$ 拟合剩下 10% 语料上的 $P_3^{10\%}(X_3|X_2, X_1)$ 。训练过程中所用的损失函数为均方误差损失：

$$L = \frac{\sum_{i=3}^{N}(M(X_i|X_{i-1}, X_{i-2}) - P_3^{10\%}(X_i|X_{i-1}, X_{i-2}))^2}{N-2} \tag{7}$$

2.3 中文信息熵的估计

在第 2.2节中提到过，可以将语言看作是一个平稳随机过程 1，且其概率分布为 P，那么中文的信息熵就可以表示为：

$$H(X) \equiv H(P) \equiv -E_P \log P(X_0|X_{-1}, X_{-2}, \ldots) = \lim_{n\to\infty} -\frac{1}{n}\log P(X_1, \ldots, X_n), \tag{8}$$

其中 $X_1, \ldots, X_n$ 是从 P 中采样得到的。

然而由于无法获取具体的 P，所以通常只能通过交叉熵来估计 $H(P)$，也就是使用 P 的一个模型 M 与 P 之间的交叉熵：

$$H(P, M) = -E_P \log M(X_0|X_{-1}, X_{-2}, \ldots) = \lim_{n\to\infty} -\frac{1}{n}\log M(X_1, \ldots, X_n), \tag{9}$$

其中 $X_1, \ldots, X_n$ 是从 P 中采样得到的。

由于信息熵与交叉熵之间有一下关系：

$$H(P) \le H(P, M). \tag{10}$$

所以可以使用交叉熵作为信息熵的上界来估计中文平均交叉熵。

3 实验

3.1 实验设置

实验使用的语料由金庸的十六部武侠小说组成，包括：《白马啸西风》，《碧血剑》，《飞狐外传》，《连城诀》，《鹿鼎记》，《三十三剑客图》，《射雕英雄传》，《神雕侠侣》，《书剑恩仇录》，《天龙八部》，《侠客行》，《笑傲江湖》，《雪山飞狐》，《倚天屠龙记》，《鸳鸯刀》，《越女剑》。在实验部分，一共进行了两组实验：一组以字为单位对语料进行切分；另一组以词（使用 jieba 工具进行中文分词）为单位对预料进行切分。对于每一组实验，分别包含 16 个实验，每个实验都以其中一部武侠小说作为测试集用以计算交叉熵，以其他武侠小说作为训练集训练语言模型，训练的方法见第 2.2节，训练过程中使用的优化算法为 Adam [2]。

此外，对于训练中不存在单元，我在验证中使用的方法是统一用一个较小的值进行替代，具体来说：

- 对于一元单元以及二元单元的频数，用 1 进行替代；
- 对于二元以及三元单元的频率，用 (1 / 单元总数) 进行替代。

3.2 实验结果

首先展示第一组实验的结果，在各个测试样本下的交叉熵，结果见表 1，可见，当以字为单位划分语料时，计算得到的平均交叉熵为 5.8273 ，最小交叉熵为 5.4386。由于交叉熵是信息上的上界，所以这里选择一个更紧的界作为对信息上的估计，也就是说在分字模式下，对中文的信息熵的估计为 5.4386 。

表 1: 以字为单元划分语料时的实验结果

测试武侠小说	训练语料总单元数	测试语料总单元数	交叉熵
《白马啸西风》	8216607	65319	5.5372
《碧血剑》	7808544	473382	5.9893
《飞狐外传》	7855148	426778	5.4567
《连城诀》	8059520	222406	5.5707
《鹿鼎记》	7111844	1170082	5.8993
《三十三剑客图》	8220907	61019	7.7268
《射雕英雄传》	7407806	874120	5.4386
《神雕侠侣》	7358707	923219	5.5868
《书剑恩仇录》	7783753	498173	5.8323
《天龙八部》	7115915	1166011	5.7434
《侠客行》	7929134	352792	5.6113
《笑傲江湖》	7340709	941217	5.6306
《雪山飞狐》	8152947	128979	5.5344
《倚天屠龙记》	7352547	929379	5.8438
《鸳鸯刀》	8248478	33448	5.6425
《越女剑》	8266324	15602	6.1937
平均值	7764306	517620	5.8273
最小值	-	-	5.4386

接着展示第二组实验的结果，在各个测试样本下的交叉熵，结果见表 2，可见，当以词为单位划分语料时，计算得到的平均交叉熵为 6.6379 ，最小交叉熵为 6.3195。所以在在分词模式下，对中文的信息熵的估计为 6.3195 。

表 2: 以词为单元划分语料时的实验结果

测试武侠小说	训练语料总单元数	测试语料总单元数	交叉熵
《白马啸西风》	5257449	43137	6.4295
《碧血剑》	5000107	300479	6.6952
《飞狐外传》	5027472	273114	6.3653
《连城诀》	5155083	145503	6.3195
《鹿鼎记》	4548437	752149	6.5986
《三十三剑客图》	5261439	39147	7.5666
《射雕英雄传》	4741095	559491	6.9699
《神雕侠侣》	4708690	591896	6.4273
《书剑恩仇录》	4984134	316452	6.9552
《天龙八部》	4550256	750330	6.5107
《侠客行》	5073765	226821	6.5581
《笑傲江湖》	4700503	600083	6.4485
《雪山飞狐》	5217546	83040	6.3605
《倚天屠龙记》	4713069	587517	6.7683
《鸳鸯刀》	5279182	21404	6.3647
《越女剑》	5290563	10023	6.8691
平均值	4969299	331287	6.6379
最小值	-	-	6.3195

从以上两个实验可以看出：

1. 分词模式下估计得到的中文信息上大于分字模式下估计的中文信息熵。也就是说，分词模式下，每个单元（token）包含的信息更高，这也符合我们的直觉；
2. 不论是分词模式还是分字模式，估计得到的中文信息熵，都比论文 [1] 中估计得到的英文信息熵（1.75）高，这也就意味着中文的每个单元所包含的信息要高于英文的每个单元所包含的信息。

3.3 消融实验

为了验证由一元、二元以及三元语言模型组成的语言模型的有效性，本文还计算了只使用三元语言模型情况下的交叉熵，实验结果见表 3。从中可以看出使用组合的语言模型，可以得到更小的交叉熵，也就是说能得到更小的对中文信息熵的估计，因此组合的语言模型能够对 P 进行更好的建模。

4 总结

本文以金庸的十六本武侠小说为语料，训练了一个将一元、二元以及三元语言模型组合的语言模型，并利用该语言模型计算得到的交叉熵，对中文的信息熵进行了估计。经过实验，本文对分字模式下的中文信息熵的

表 3: 组合语言模型与三元语言模型的比较

测试武侠小说	分字		分词	
	组合语言模型	三元语言模型	组合语言模型	三元语言模型
《白马啸西风》	5.5372	6.8318	6.4295	10.5413
《碧血剑》	5.9893	7.4149	6.6952	11.2886
《飞狐外传》	5.4567	6.5761	6.3653	10.6898
《连城诀》	5.5707	6.6980	6.3195	10.3940
《鹿鼎记》	5.8993	7.1896	6.5986	10.9221
《三十三剑客图》	7.7268	9.9568	7.5666	13.2298
《射雕英雄传》	5.4386	6.6255	6.9699	10.7995
《神雕侠侣》	5.5868	6.7671	6.4273	10.8527
《书剑恩仇录》	5.8323	7.2019	6.9552	11.2094
《天龙八部》	5.7434	7.1287	6.5107	10.8154
《侠客行》	5.6113	6.5719	6.5581	10.4685
《笑傲江湖》	5.6306	6.8151	6.4485	10.6237
《雪山飞狐》	5.5344	6.6125	6.3605	10.6733
《倚天屠龙记》	5.8438	7.1278	6.7683	10.9438
《鸳鸯刀》	5.6425	6.7527	6.3647	10.5291
《越女剑》	6.1937	7.4944	6.8691	11.4536
平均值	5.8273	7.1103	6.6379	10.9647
最小值	5.4386	6.5719	6.3195	10.3940

估计值为 5.4386，对分词模式下的中文信息熵估计为 6.3195。此外，本文还通过消融实验验证了组合语言模型的有效性。

参考文献

[1] P. F. Brown, S. D. Pietra, V. J. D. Pietra, J. C. Lai, and R. L. Mercer, "An Estimate of an Upper Bound for the Entropy of English," *Comput. Linguistics*, vol. 18, no. 1, pp. 31–40, 1992.

[2] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.