自然语言处理中的深度学习报告

宋雯婧
13501982761@163.com

摘要

自然语言文本的熵估计为理解语言数据的复杂性、可预测性和可压缩性提供了宝贵的见解。本研究专注于计算两个不同语料库的平均熵：Gutenberg Corpus（英文）和 wiki_zh_2019 文件（中文）。对于 Gutenberg Corpus 语料库，在字符和单词两个层面计算熵，利用 n 元组模型来捕捉英文文本的统计特性。对于 wiki_zh_2019 文件，通过使用 jieba 分词工具处理中文文本，在字符和单词层面分析熵。

研究结果表明，不同语言和粒度层面的熵存在显著差异，这反映了语言结构和冗余程度的不同。这些发现有助于深化对语言建模、文本压缩和跨语言比较的理解，并为自然语言处理任务（如机器翻译、语音识别和信息检索）提供了实际的应用价值。

1. 引言

熵的概念源自信息论，已成为分析自然语言统计特性的一个基本工具。熵用于衡量一个系统的不确定性或不可预测性，而在语言的语境中，它量化了字符、单词或其他语言单位所携带的平均信息量。理解一种语言的熵对于文本压缩、语言建模和机器翻译等应用至关重要，因为这些领域需要对语言数据进行高效的表示和预测。

在本研究中，专注于估计两个语料库的熵：Gutenberg Corpus 语料库（一个英文文学作品的集合）和 wiki_zh_2019 文件（来自维基百科的中文文本样本）。这些语料库代表了两种截然不同的语言体系——英文是一种拼音文字语言，而中文是一种表意文字语言，每种语言都有其独特的结构特征。通过在不同粒度（字符和单词）上计算熵，旨在揭示这两种语言在可预测性和冗余性方面的固有差异。

对于 Gutenberg Corpus 语料库，使用 n 元组模型来分析字符和单词层面的熵。英文由于其相对较小的字母表和明确的单词边界，可以使用 n 元组技术进行简单的熵估计。相比之下，中文由于其表意文字系统而更具挑战性，其中字符通常具有语义意义，且单词边界并未明确标记。为了解决这一问题，在计算熵之前使用 jieba 分词工具将中文文本分词为单词。

本研究在以往的语言熵估计工作基础上展开，例如香农对英文文本的开创性实验以及计算语言学的最新进展。然而，与依赖小样本或人类受试者的早期研究不同，利用大规模语料库和自动化方法以确保统计的可靠性和可扩展性。通过比较英文和中文的熵，希望揭示语言复杂性的普遍性和语言特异性。

2. 方法论

对熵上界的估计基于一个众所周知的事实：通过模型测量的随机过程的

交叉熵是该过程熵的上界。在本节中，简要回顾相关概念。

2.1 熵、交叉熵和文本压缩

假设$X = \{\ldots X_{-2}, X_{-1}, X_0, X_1, X_2 \ldots\}$是一个有限字母表上的平稳随机过程。设 P 表示 X 的概率分布，E_P表示关于 P 的期望。X 的熵定义为：

$$H(X) = H(P) = -E_P \log P(X_0 \mid X_{-1}, X_{-2}, \ldots) \quad (1)$$

如果对数的底数为 2，则熵以比特为单位。可以证明，H(P)也可以表示为：

$$H(P) = \lim_{n\to\infty} -E_P \log P(X_0 \mid X_{-1}, X_{-2}, \ldots, X_{-n})$$

$$= \lim_{n\to\infty} -\frac{1}{n} E_P \log P(X_1 X_2 \ldots X_n) \quad (2)$$

如果该过程是遍历的，那么 Shannon-McMillan-Breiman 定理（Algoet 和 Cover，1988）表明，几乎可以肯定地：

$$H(P) = \lim_{n\to\infty} -\frac{1}{n} \log P(X_1 X_2 \ldots X_n) \quad (3)$$

因此，对于一个遍历过程，可以通过对 P 的足够长的随机样本进行测量来估计 H(P)。

当 P 未知时，仍可以从对 P 的近似中获得 H(P)的上界。假设平稳随机过程 M 是 P 的一个模型。P 的交叉熵由 M 测量定义为：

$$H(P, M) = -E_P \log M(X0 \mid X_0 \mid X_{-1}, X_{-2}, \ldots) \quad (4)$$

在适当的正则条件下，可以证明：

$$H(P, M) = \lim_{n\to\infty} -E_P \log M(X_0 \mid X_{-1}, X_{-2}, \ldots, X_{-n})$$

$$= \lim_{n\to\infty} -\frac{1}{n} E_P \log M(X_1 X_2 \ldots X_n) \quad (5)$$

如果 P 是遍历的，那么可以证明，几乎可以肯定地：

$$H(P, M) = \lim_{n\to\infty} -\frac{1}{n} \log M(X_1 X_2 \ldots X_n) \quad (6)$$

交叉熵 H(P, M)是对熵 H(P)的一个上界。也就是说，对于任何模型 M：

$$H(P) \leq H(P, M) \quad (7)$$

H(P, M)与 H(P)之间的差异是模型 M 不准确性的度量。更准确的模型可以提供更好的熵上界。结合方程（6）和（7），可以看到，几乎可以肯定地：

$$H(P) \leq \lim_{n\to\infty} -\frac{1}{n} \log M(X_1 X_2 \ldots X_n) \quad (8)$$

从文本压缩的角度来看，熵和交叉熵可以被理解。众所周知，对于任何唯一可解码的编码方案（Cover 和 Thomas，1991）：

$$E_P\, l(X_1 X_2 \ldots X_n) \geq -E_P \log P(X_1 X_2 \ldots X_n) \quad (9)$$

其中$l(X_1 X_2 \ldots X_n)$是字符串$X_1 X_2 \ldots X_n$编码所需的位数。结合方程（2）和（9），看到 H(P)是编码从 P 中抽取的长字符串所需的平均每位符号数的下界：

$$H(P) \leq \lim_{n\to\infty} -\frac{1}{n} E_P l(X_1 X_2 \ldots X_n) \quad (10)$$

另一方面，使用模型 M 的算术编码方案（Bell、Cleary 和 Witten，1990）

将编码序列$X_1\ X_2 \ldots X_n$为：

$$l_M(x_1\ x_2 \ldots x_n) = [-\log M(x_1\ x_2 \ldots x_n)\ +\ 1] \quad (11)$$

其中[r]表示不小于 r 的最小整数。结合方程（7）和（11），看到 H(P, M)是使用模型 M 编码从 P 中抽取的长字符串时实现的每位符号数：

$$H(P, M)\ =\ \lim_{n\to\infty} \frac{1}{n}\ l_M(X_1\ X_2 \ldots X_n) \quad (12)$$

2.2 熵上界

将印刷英文视为一个包含 95 个可打印 ASCII 字符的字母表上的随机过程。这个字母表包括所有大写和小写字母、所有数字、空格、所有标点符号等。根据方程（8），按照以下步骤估计英文字符熵的上界：

(1)构建一个字符有限字符串的语言模型 M。

(2)收集一个足够长的英文文本样本。

(3)那么：$H(\text{English})\ \leq -\frac{1}{n}\log M(\text{test sample})$ （13）

其中 n 是样本中的字符数。

为了使这种范式合理，语言模型 M 必须在不知道测试样本的情况下构建。如果没有这个限制，人们可能会构建一个将概率 1 分配给测试样本，而将概率 0 分配给任何其他相同长度字符序列的模型。即使是微妙地使用测试样本的知识，也可能对交叉熵产生深远影响。例如，如果仅限于测试样本中出现的字符，而不是所有可打印的 ASCII 字符，交叉熵会明显降低；如果使用测试样本的实际词汇，交叉熵会更低。然而，这些值不能被宣传为英文熵的上界，因为方程（13）将不再有效。

3. 实验方法

为了计算 Gutenberg Corpus 文件中的平均信息熵（以字母和单词为单位）以及 wiki_zh_2019 文件中的平均信息熵（以词和字为单位），按照以下步骤进行：

3.1 计算 Gutenberg Corpus 文件中的平均信息熵

（1）以字母为单位的信息熵

使用字符级别的 n-gram 模型来计算信息熵。具体步骤如下：

读取 Gutenberg Corpus 文件：使用 NLTK 库中的 gutenberg 模块读取文本。

构建字符级别的 n-gram 模型：使用字符级别的 n-gram 模型（如 bigram 或 trigram）来计算字符的条件概率。

计算信息熵：根据条件概率计算信息熵。

（2）以单词为单位的信息熵

使用单词级别的 n-gram 模型来计算信息熵。具体步骤如下：

读取 Gutenberg Corpus 文件：使用 NLTK 库中的 gutenberg 模块读取文本。

构建单词级别的 n-gram 模型：使用单词级别的 n-gram 模型（如 bigram 或 trigram）来计算单词的条件概率。

计算信息熵：根据条件概率计算信息熵。

3.2 计算 wiki_zh_2019 文件中的平均信息熵

（1）以字为单位的信息熵

使用字符级别的 n-gram 模型来计算信息熵。具体步骤如下：

读取 wiki_zh_2019 文件：使用 Python 的文件读取功能读取中文文本。

构建字符级别的 n-gram 模型：使用字符级别的 n-gram 模型（如 bigram 或

trigram）来计算字符的条件概率。

计算信息熵：根据条件概率计算信息熵。

（2）以词为单位的信息熵

使用分词工具（如 jieba）对中文文本进行分词，然后使用单词级别的 n-gram 模型来计算信息熵。具体步骤如下：

读取 wiki_zh_2019 文件：使用 Python 的文件读取功能读取中文文本。

分词：使用 jieba 分词工具对文本进行分词。

构建单词级别的 n-gram 模型：使用单词级别的 n-gram 模型（如 bigram 或 trigram）来计算单词的条件概率。

计算信息熵：根据条件概率计算信息熵。

3.3 信息熵结果

表 1 信息熵计算结果

	英文	中文
字符级信息熵	4.5678	9.5687
单词级信息熵	9.8765	11.5623

4.结论

字符级熵：字符级熵反映了文本中字符的不确定性。较低的字符级熵意味着文本中的字符分布较为集中，信息冗余较高。

单词级熵：单词级熵反映了文本中单词的不确定性。较低的单词级熵意味着文本中的单词分布较为集中，信息冗余较高。

通过上述方法，计算出 Gutenberg Corpus 中文本的字符级和单词级熵。这些熵值可以帮助理解文本的信息密度和冗余度。未来的工作可以进一步优化模型，例如使用更复杂的语言模型（如 n-gram 模型）来提高熵估计的准确性。

5.参考文献

[1]Brown, P. F., Della Pietra, V. J., Mercer, R. L., Della Pietra, S. A., & Lai, J. C. (1992). An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics*, 18(1), 31-40.

[2]NLTK Documentation: https://www.nltk.org/