

深度学习与自然语言处理课程报告

——基于 N 元语言模型的中文信息熵计算

姓名：龙行健

学号：ZY2203110

一、摘要

信息熵在机器学习中被用来衡量一个系统的信息含量，如果一个系统越复杂，出现不同情况的种类越多，那么其信息熵是较大的，反之其信息熵较小。拥有 26 个基础字母的英语信息熵已经被衡量大约为 1.75 比特/符左右，而中文的基础字符要比英语多得多，其信息熵应比英语要高。为了衡量中文字词的具体信息含量，本实验通过采用不同的 N 元语言模型，基于金庸小说数据集，统计了中文字词的信息熵。实验结果表明，采用 1、2、3 元语言模型的统计中，中文具有较大的信息熵，但随着语言模型元数的增加，中文的信息熵在减小。

二、引言

香农于 1948 年借鉴热力学中的“熵”的概念，提出了信息熵，其被用来衡量一个系统的信息含量多少。熵值越大，那么系统的单位信息含量就越大。其数学公式计算可表示为 [1]:

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

$$X_0 \in X, X = \{\dots, X_{-2}, X_{-1}, X_0, X_1, X_2, \dots\}$$

其中， X 为基于有限字符取值形成的一平稳随机过程。将其通过一定的变换写成常见的形式如下：

$$H(X) = -\sum P(X) \log P(X)$$

基于某种语言模型，可以将上式写为：

$$H(X) = -E_p \log M(X_0 | X_{-1}, X_{-2}, \dots, X_{-n})$$

如此，我们就可以将一个未知的概率模型 P 近似用一个语言概率模型 M 来进行衡量，而这个语言模型中的变量可以根据假设简化为有限固定个数，进而对某种语言的信息熵进行估计。

三、 实验原理

1. N 元语言模型

一元模型的信息熵计算公式为：

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

其中， \log 是以 2 为底的对数， $p(x)$ 为每个词在一元语料库中出现的频率。

当一个词出现得频率越高，那么其频率就越大，其期望也会越大，进而表明这个词在语料库中很重要。但是很重要或许又和含信息量多有些微的矛盾，比如中文的助词，像是“的”、“其”等，其出现的频率固然高，但是实际的含义并不高，所以具体的计算还需做一些处理。

对于更高维度的语言模型，其计算公式为：

$$\begin{aligned} H(X) &= - \sum_{x \in X} p(x_0, x_{-1}, \dots, x_{-n}) \log p(x_0 | x_{-1}, \dots, x_{-n}) \\ &= - \sum_{x \in X} p(x_0, x_{-1}, \dots, x_{-n}) \log \frac{p(x_0, x_{-1}, \dots, x_{-n})}{p(x_{-1}, \dots, x_{-n})} \end{aligned}$$

其中，第二个等号根据的是条件概率公式。

上述模型的依据是某个单词出现的概率和其前 N 个单词的组合有关，对于不同的组合，导致某个单词在该位置出现的概率也会变化。这是符合直觉的，例如跟在“吃”这个字后面出现的词，通常认为是某一种食物，“吃桌子”出现概率是较小的。将上述的模型作为某个字词出现的信息概率衡量，可以完成对于中文信息熵的估计。

本文基于如下假设进行信息熵计算：

- 1) 每个字词不仅和本句话有关联，也和上一个句子中的字词存在关联；
- 2) 每个字词只和前 N 个字词的出现存在关联；

四、 实验过程

1. 实验环境

带有 jieba 开源库和基础科学运算的 python 环境

2. 数据来源

1) 统计数据集

金庸武侠小说 16 部。

2) 停词表

包括非中文字符，本次实验中没有用上无实义助词作为停词表。

3. 数据预处理

这部分主要对金庸武侠小说数据集进行三个操作：数据集读取、去除不计入统计的词汇、采用 jieba 中文语料库进行分词操作。

```
def Preprocess(data_root, del_root, aft_del_root):
    data_list_dir = os.listdir(data_root)
    del_list_dir = os.listdir(del_root)
    aft_del_list_dir = os.listdir(aft_del_root)
    data_corpus = []
    del_corpus = []
    aft_del_corpus = []
    cha_count = 0

    #First preprocess
    for del_file_name in del_list_dir:
        del_file_path = del_root + '/' +str(del_file_name)
        print(del_file_path)
        with open(os.path.abspath(del_file_path), "r", encoding = 'utf-8') as f:
            del_context = f.read()
            del_corpus.extend(del_context.split("\n"))
    for data_file_name in data_list_dir:
        data_file_path = data_root + '/' +str(data_file_name)
        print(data_file_path)
        with open(data_file_path, "r", encoding = 'ANSI') as f:
            data_context = f.read()
            data_context = data_context.replace("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", '')
            data_context = data_context.replace("\n", "")
            data_context = data_context.replace(" ", '')
            data_context = re.sub('\s', '', data_context)

            for del_word in del_corpus:
                data_context = data_context.replace(del_word, "")
            cha_count += len(data_context)
            data_corpus.append(data_context)

    data_corpus = jiebaCut(data_corpus)
def jiebaCut(corpus):
    new_corpus = []
    for text in corpus:
        words = jieba.cut(text)
        new_corpus.extend(words)
    return new_corpus
```

4. N 元语言模型处理

首先计算 N 元语料库中每一个元组合出现的频率：

```
def nWordsFre(corpus, N):
    dic_fre = {}
    if N <= len(corpus):
        for i in range(len(corpus)-N):
            tempt = (corpus[i])
            for j in range(1,N):
                tempt = tempt, (corpus[i+j])
            dic_fre[tempt] = dic_fre.get(tempt, 0) + 1
    else:
        print("Error N number.")
    return dic_fre
```

计算不同 N 元语料库模型得到的数据集信息熵，N 元模型只用计算 N 元下的语料库和 N-1 元的语料库即可：

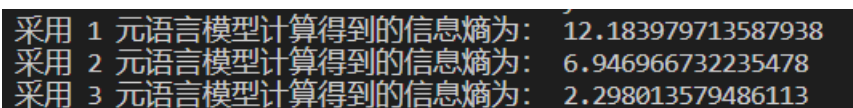
```
def calNWordsEntropyModel(corpus, N):
    #Input:N(Positive Integer)
    #Output:N-element model entropy
    if N == 1:
        dic_fre = nWordsFre(corpus, 1)
        n_len = 0
        entropy = 0
        for item in dic_fre.items():
            n_len += item[1]
        for item in dic_fre.items():
            entropy += -item[1] / n_len * math.log(item[1] / n_len, 2)
    else:
        dic_fre_n = nWordsFre(corpus, N)
        dic_fre_n_1 = nWordsFre(corpus, N-1)
        n_len = 0
        entropy = 0
        for item in dic_fre_n.items():
            n_len += item[1]
        # If data base is very big, then n_len equal to n_1_len
        for item in dic_fre_n.items():
            p_joint = item[1] / n_len
            p_cond = item[1] / dic_fre_n_1[item[0][0]]
            entropy += -p_joint * math.log(p_cond, 2)
    return entropy
```

五、 实验结果

1、2、3 三个不同的预料模型得到的信息熵如图 1.所示。可以看出，到 3 元模型为止，中文的信息熵都是显然大于英文的信息熵的，但是其有逐渐下降的趋势。其实这也是容易理解的，因为随着 N 元组的增大，每个元组出现多次的难度越来越高，其对应的频率也慢慢趋于一致，极限情况如下所示：

$$H(X) = \log \frac{Num}{N}$$

当 N 增大到和语料库的字词总数相等时，最终的信息熵为 0，这时一个一元组就可以囊括整个信息传递，其信息含量当然是没有的。



```
采用 1 元语言模型计算得到的信息熵为: 12.183979713587938
采用 2 元语言模型计算得到的信息熵为: 6.946966732235478
采用 3 元语言模型计算得到的信息熵为: 2.298013579486113
```

图 1.三种不同语言模型计算得到的信息熵

六、 结论

上述实验过程和结果表明，中文的信息熵在基于 N 元语料模型下，在小 N 的情况下是要比英语的信息熵要高的，这反映出中文字词的紧密联合性。当 N 为 1 的时候，中文的信息熵最高，在这种情况下考虑下，中文的字词频率分布不均匀，并且每个中文字符的信息含量都很高。

七、 参考文献

- [1] *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.*