熵计算实验报告

陈江昊 人工智能学院 202228014628016 代码地址

实验介绍

1. 利用爬虫工具从互联网上收集大量的中英文文本数据，并对收集到的数据进行清洗；
2. 设计算法并编程实现在收集文本数据上中文汉字和英文字母的概率和熵的计算；
3. 改变文本数据规模，重新计算中文汉字和英文字母的概率和熵，并分析计算结果。

实验数据收集

中文数据

根据URL以及HTML标签的规律，对百度百科不同词条下网页中的文本数据进行爬取。百度百科的URL形式为`"https://baike.baidu.com/view/" + str(i) + ".htm"`，规律性强，从而能够方便的获取大量的URL链接。此外，利用`BeautifulSoup`库对网页HTML源码进行分析，从中提取中文文本数据。具体代码实现如下：

```
def get_text_ch(url, num, save_path, save_feq=100):
    os.makedirs(save_path, exist_ok=True)
    dic_ch = {}
    for idx in tqdm(range(1, num + 1, 1)):
        try:
            r = requests.get(url + str(idx) + '.htm', headers=headers,
proxies=proxies)
            # r.encoding = r.apparent_encoding
            soup = BeautifulSoup(r.text, 'html.parser')
            text = ''
            for x in soup.find_all('div', {'class': 'para'}):
                text += x.text
            dic_ch[url + str(idx) + '.htm'] = text
            r.close()
        except:
            pass

        if idx % save_feq == 0:
            u = list(dic_ch.keys())
            t = list(dic_ch.values())
            result = pd.DataFrame()
            result["url"] = u
            result["text"] = t
            result.to_excel(os.path.join(save_path, "./data{}.xlsx".format(idx)))
            dic_ch = {}
```

英文数据

由于未能找到URL较为规律的英文网站，我们决定对 Wikipedia 上的数据进行爬取，并以 Wikipedia 上某个词条的网页为种子，寻找到该网页中所有链接到 Wikipedia 其他词条的超链接，然后对找到的新链接再进行数据爬取，以此类推。具体代码实现如下：

```
def get_text_en(url, num, keyword, save_path, save_feq=100):
    os.makedirs(save_path, exist_ok=True)
    mem = set()
    mem.add("/wiki/" + keyword)
    dic_en = {}
    for idx in tqdm(range(1, num + 1, 1)):
```

```
        key = mem.pop()
        try:
            r = requests.get(url + key, headers=headers, proxies=proxies)
            # r.encoding = r.apparent_encoding
            soup = BeautifulSoup(r.text, 'html.parser')
            text = ''
            for x in soup.find_all('p'):
                text += x.text
            cnt = 0
            link = soup.find_all('a')
            # random.shuffle(link)
            for x in link:
                tmp = x.get('href')
                if tmp and tmp.startswith('/wiki/') and tmp not in mem:
                    mem.add(tmp)
                    cnt += 1
                    if cnt == 3:
                        break
            dic_en[url + key] = text
            r.close()
        except:
            pass
        if idx % save_feq == 0:
            u = list(dic_en.keys())
            t = list(dic_en.values())
            result = pd.DataFrame()
            result["url"] = u
            result["text"] = t
            result.to_excel(os.path.join(save_path, "./data{}.xlsx".format(idx)))
            dic_en = {}
```

实验数据清洗

由于本实验只需要对中文汉字和英文字母进行相关计算，我们只需要使用正则表达式相关的库`re`过滤掉中文数据里的非汉字，统一英文数据的大小写，并且过滤到其中的非英文字母部分即可。最终清洗完毕，得到数据的样本规模约为：7×10^7个中文汉字，2×10^8个英文字母。

实验内容与结果分析

根据离散随机变量熵的公式$H(x) = -\sum_{i=1}^{n} p(x_i)log_2p(x_i)$，我们利用`collection`库函数`Counter`统计中文汉字和英文字母出现的频率，构建一个键为汉字/字母，值为频率的字典，并用频率近似公式中的概率，便能够计算得到最终的熵。此外，为了分析数据规模对熵计算结果的影响，我们逐步扩大文本规模进行实验。具体代码实现如下：

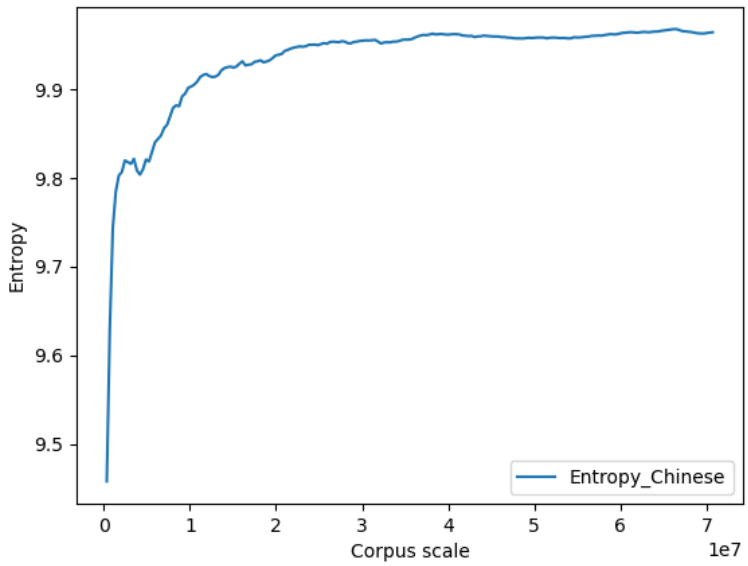
```
# calculate entropy
def cal_h(dic, n):
    h = 0
    for v in dic.values():
        h += - v / n * math.log(v / n, 2)
    return h

# Chinese
for txt in tqdm(ch_txt_list):
    if pd.isna(txt):
        continue
    txt_rm = re.sub(r'[^\u4e00-\u9fa5]+', '', txt)
    ch_txt_list_processed.append(txt_rm)
ch_N = len("".join(ch_txt_list_processed))

```

```python
ch_scale = []
ch_H = []
for num in tqdm(range(350000, ch_N + 1, 350000)):
    random.shuffle(ch_txt_list_processed)
    txt_data = "".join(ch_txt_list_processed)[:num]
    d = dict(Counter(txt_data))
    H_Chinese = cal_h(d, num)
    ch_scale.append(num)
    ch_H.append(H_Chinese)

# English
for txt in tqdm(en_txt_list):
    if pd.isna(txt):
        continue
    try:
        txt_rm = re.sub(r"[^\u0041-\u005a\u0061-\u007a]+", "", txt)
        txt_rm = txt_rm.lower()
        en_txt_list_processed.append(txt_rm)
    except:
        print(txt)
en_N = len("".join(en_txt_list_processed))

en_scale = []
en_H = []

for num in tqdm(range(1000000, en_N, 1000000)):
    random.shuffle(en_txt_list_processed)
    txt_data = "".join(en_txt_list_processed)[:num]
    d = dict(Counter(txt_data))
    H_English = cal_h(d, num)
    en_scale.append(num)
    en_H.append(H_English)
```

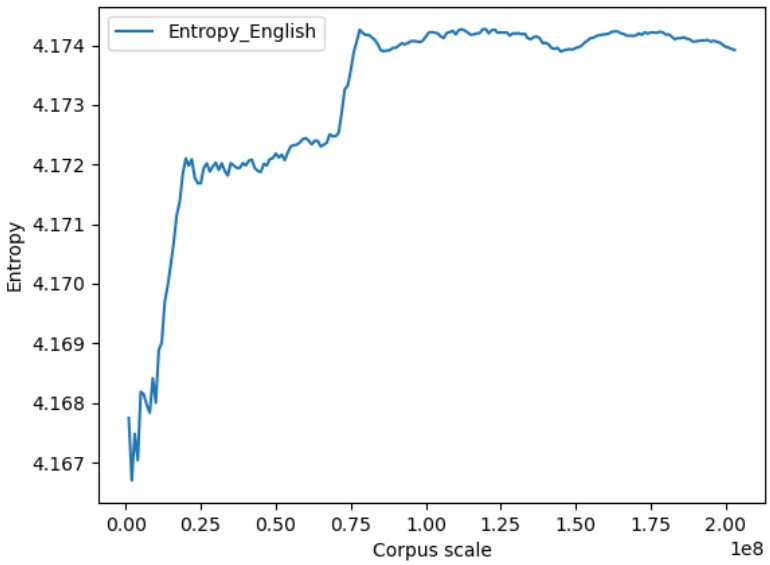
最终计算得到中文汉字和英文字母的熵如下：

$H_{Chinese} = 9.96$，$H_{English} = 4.17$

实验中，我们首先按收集到文本的顺序，每次增加总数据量的$\frac{1}{20}$，得到如下的熵随文本数据规模的变化曲线：

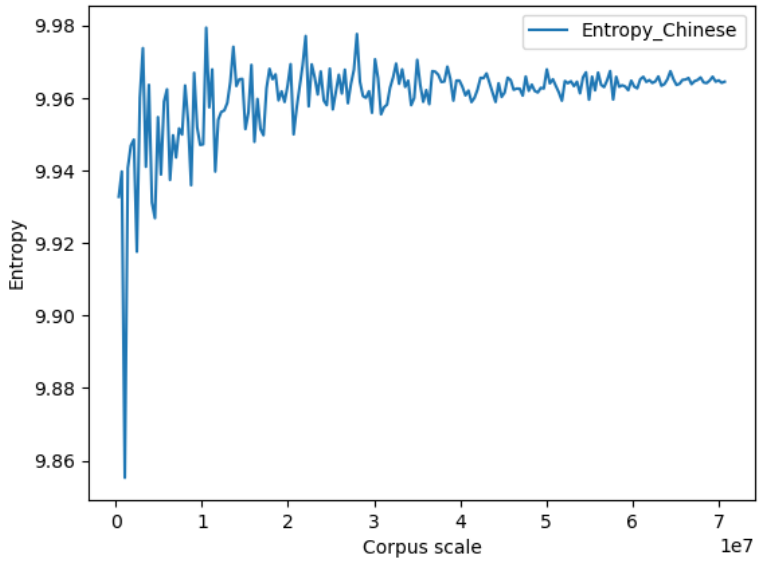


中文汉字的熵随文本数据规模扩大的变化曲线

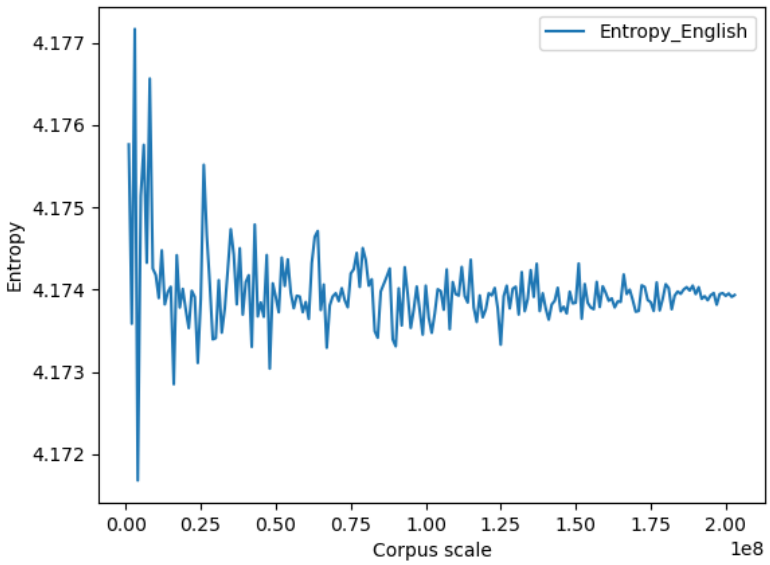


英文字母的熵随文本数据规模扩大的变化曲线

可以看出随着文本数据的增大，熵的值逐渐增加，最终呈现出收敛的趋势，这符合大数定理中频率趋近于概率的现象。此外，我们发现熵在数据规模刚开始扩大时有一个明显的上升趋势，而且整个过程波动较小。我们猜测这个现象与我们的数据收集方式有关。例如，我们的英文数据是通过 Wikipedia 词条之间的相互链接所爬取到的。因此，一开始收集到的数据之间相关性可能较大，从而熵较小。而随着数据规模的扩大，词条之间的相关性减弱，熵就呈现一个上升的趋势。我们的中文数据是通过URL链接中的ID每次加一爬取到的，相邻ID之前可能也有着较大的相关性。为了验证这一点，我们将原本的按照爬取顺序增加数据规模的方式改成随机打乱筛选的方式，得到如下结果：



中文汉字的熵随文本数据规模扩大的变化曲线



英文字母的熵随文本数据规模扩大的变化曲线

对比两次实验结果，我们发现在加入随机打乱筛选的操作后，熵的值波动增大，上升趋势减弱，并且依然呈现了收敛的趋势。