自然语言处理第一次作业（A）

作者：张宗桁
学号：2023K8009991013

2025 年 11 月 8 日

目录

1 样本的爬取过程

本次实验利用 Python 语言的 Scrapy 框架进行网络数据的爬取。Scrapy 是一个为了爬取网站数据、提取结构性数据而编写的高效、异步的应用框架，它允许开发者通过定义“爬虫”（Spiders）来实现复杂的抓取逻辑。

本实验的核心爬取逻辑由两个 Spider 类（'arxiv_spider.py' 和 'sina_tech.py'）定义，它们分别针对 $arXiv$ 和新浪新闻这两个数据源。

1.1 $arXiv$ 论文数据爬取 (arxiv_spider.py)

$arXiv$ 数据源的爬取目标是获取计算机科学（CS）领域（包括人工智能 'cs.AI'、计算语言学 'cs.CL' 和计算机视觉 'cs.CV'）的最新论文文本：

1. **起始点 (parse 方法):** 爬虫从 `start_urls` 中定义的 `/recent` 列表页开始。在此页面，它使用 CSS 选择器 (`response.css`) 查找所有指向论文摘要页（`/abs/...`）的链接，并使用 `response.follow` 生成新的请求，将响应(response)交由 `parse_abstract` 方法处理。
2. **摘要爬取 (parse_abstract 方法)**
   - 爬虫执行如下方案：仅提取当前页面的论文摘要。摘要文本位于 `<blockquote>` 标签中。
3. **内容提取 (parse_html_paper / parse_abstract):** 在 `parse_abstract` 方法中提取摘要内容。

1.2 新浪新闻数据爬取 (sina_tech.py)

`sina_tech.py` 爬虫的逻辑相对直接。它从首页开始，在 `parse` 方法中使用 XPath (`response.xpath`) 筛选出所有符合新浪文章 URL 特征（包含 `/doc-` 和 `.shtml`）的链接。随后，它跟进这些链接，并在 `parse_article` 方法中，使用 XPath 精确提取文章的标题、发布时间、来源和正文等结构化数据。

1.3 核心爬取配置 (settings.py)

为了确保爬取过程的高效、稳定和友好，在 `settings.py` 文件中进行如下参数设置：

1. **友好性策略：** 为了避免对目标服务器造成过大压力，组合使用 `ROBOTSTXT_OBEY = True`（严格遵守网站的 `robots.txt` 爬取协议）和 `DOWNLOAD_DELAY = 1`（设置 1 秒的下载延迟）。
2. **身份伪装：** 将 `USER_AGENT` 设置为通用的 Chrome 浏览器标识，以模拟正常的用户访问，这是爬虫反屏蔽的常见策略。
3. **数据编码：** 鉴于新浪新闻包含大量中文文本，明确设置 `FEED_EXPORT_ENCODING = "utf-8"`。

2 样本的清洗

数据清洗是本实验至关重要的一步，其目的是从爬取原始文本中提取出干净、可用的纯文本内容。我们主要使用了 Python 的 BeautifulSoup 库和正则表达式 (re) 库。

2.1 中文处理

对于新浪新闻的中文文本，我们进行了以下清洗步骤：

1. **移除特定噪音：** 移除特定的广告（如炒股就看...）、系统残留（如 Flash Player...）和免责声明等。
2. **移除 URL 和邮件：** 移除多种格式的作者、来源、责编等信息（如（来源：...）或（责编|记者...））。
3. **去除无效字符：** 去除多余的空白符、换行符和特殊控制字符。
4. **统一编码：** 所有文本统一保存为 UTF-8 编码。

2.2 英文处理

对于 $arXiv$ 论文的英文文本，清洗步骤包括：

1. **移除元数据：** 移除特定的 `\s*` 标签。
2. **文本标准化：** 使用 `cleaned_line.lower()`将所有文本转换为小写。
3. **移除格式化残留：** 移除如 `textbf` 或 `textit` 这样的格式前缀。
4. **严格字符过滤：** 移除所有非字母和非空白字符（即数字和标点）。
5. **规范化空白：** 将多个连续空白压缩为单个空格。
6. **去重与写入：** 使用 `set` 集合 (`seen_lines`) 检查，跳过已存在的行。

最终，我们得到了三个不同规模的中文和英文语料库：

- **小规模样本：** 中文约 150KB 文本，英文约700KB。
- **中规模样本：** 中文约 250KB 文本，英文约900KB。
- **大规模样本：** 中文约 350KB 文本，英文约1500KB。

3 英语单词或汉字的概率和熵

在此部分，我们分别对中文和英文样本进行信息熵的计算。信息熵的计算公式为：

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

其中 $P(x_i)$ 是一个符号（单词或汉字）出现的概率。

3.1 中文

对于中文，我们以“汉字”为单位进行统计。

3.1.1 结果

排名	汉字	次数	概率
1	的	1503	2.7301%
2	在	531	0.9645%
3	为	449	0.8156%
4	一	435	0.7902%
5	能	366	0.6648%
6	中	359	0.6521%
7	是	309	0.5613%
8	了	308	0.5595%
9	大	302	0.5486%
10	成	281	0.5104%
11	人	272	0.4941%
12	年	268	0.4868%
13	同	263	0.4777%
14	业	258	0.4686%
15	发	258	0.4686%
16	有	256	0.4650%
17	用	245	0.4450%
18	新	244	0.4432%
19	国	239	0.4341%
20	品	239	0.4341%

图 1: 小规模样本的汉字频率和信息熵

排名	汉字	次数	概率
1	的	2186	2.6710%
2	在	690	0.8431%
3	业	637	0.7783%
4	一	608	0.7429%
5	为	578	0.7062%
6	年	566	0.6916%
7	不	557	0.6806%
8	是	555	0.6781%
9	行	491	0.5999%
10	中	490	0.5987%
11	发	470	0.5743%
12	人	441	0.5388%
13	生	441	0.5388%
14	有	434	0.5303%
15	能	426	0.5205%
16	上	414	0.5058%
17	大	389	0.4753%
18	了	381	0.4655%
19	国	379	0.4631%
20	学	368	0.4496%

图 2: 中规模样本的汉字频率和信息熵

排名	汉字	次数	概率
1	的	3382	3.1309%
2	在	1047	0.9693%
3	一	901	0.8341%
4	是	855	0.7915%
5	业	854	0.7906%
6	年	757	0.7008%
7	为	688	0.6369%
8	大	635	0.5878%
9	不	608	0.5629%
10	有	607	0.5619%
11	中	590	0.5462%
12	能	581	0.5379%
13	人	574	0.5314%
14	了	561	0.5193%
15	上	550	0.5092%
16	资	538	0.4981%
17	理	496	0.4592%
18	场	474	0.4388%
19	产	466	0.4314%
20	成	460	0.4258%

图 3: 大规模样本的汉字频率和信息熵

3.1.2 横向对比

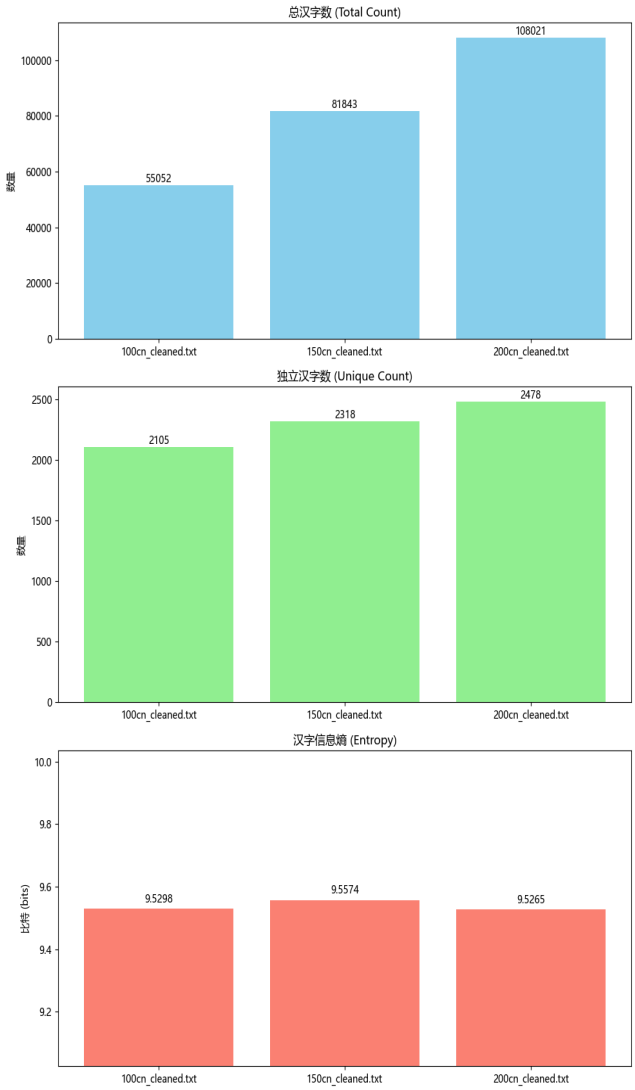

图 4: 横向对比

分析 根据图表数据分析，`100cn_cleaned.txt`、`150cn_cleaned.txt` 和 `200cn_cleaned.txt` 三个文本文件的汉字信息熵变化呈现先升后降（9.5298 → 9.5574 → 9.5265）的非线性趋势。

这一现象清晰地表明，影响信息熵的核心因素是文本的“主题内容”及其决定的“汉字概率分布”，而非文本规模本身。下面我们分阶段进行详细论证：

第一阶段：从 100cn 到 150cn（熵的合理增长）

1. **数据表现：** 在此阶段，总汉字数（55,052 → 81,843）、独立汉字数（2,105 → 2,318）和信息熵（9.5298 → 9.5574）均呈现增长。

2. **分布变化：** 这是一个符合预期的增长。文本规模扩大，带来了 213 个新的独立汉字，引入了更多样化的词汇和表达。关键证据是，最常用字“的”的概率反而轻微下降了（从 2.7301% → 2.6710%）。

3. **结论**：当一个文本库在扩展时，如果新加入的内容主题多样，会引入新的词汇（独立汉字增加），同时稀释最常见词汇的占比。这种多样化使得汉字分布更趋于均匀，系统的不确定性随之增加，因此信息熵上升。

第二阶段：从 150cn 到 200cn（熵的反常下降）

1. **数据表现**：在此阶段，总汉字数（81,843 → 108,021）和独立汉字数（2,318 → 2,478）依然在增加，但信息熵却反常下降了（从 9.5574 → 9.5265）。

2. **分布变化**：熵下降的直接原因是汉字概率分布变得更不均匀。最显著的证据是：

   - 排名第一的汉字“的”，其出现概率从 2.6710% **显著飙升**至 3.1309%。
   - 排名第二的“在”，概率也从 0.8431% 上升到 0.9693%。

3. **理论解释**：根据信息熵的定义 $H = -\sum p(x) \log p(x)$，当某个单一事件（如“的”字）的概率 p 显著提高时，它对总熵的贡献会减小，导致整体系统的不确定性（熵）降低。换言之，当“的”字出现得如此频繁，系统就变得“更可预测”了。

4. **主题偏移**：词频表（Top 20）的变化揭示了熵下降的根本原因——“主题偏移”。

   - `150cn` 的高频词包括“行”（第9）、“生”（第13）、“学”（第20）。
   - `200cn` 中，这些词跌出前20，取而代之的是一组主题高度集中的新词：“资”、“理”、“场”、“产”。

5. **结论**：这表明从 150cn 到 200cn 新增的文本内容，大概率是高度集中在“商业”、“经济”或“管理”领域。这种高度专业化、主题趋同的文本，不仅引入了自己领域的高频词，也导致了“的”、“在”等通用字的用法频率激增。这种“主题收敛”（Topical Convergence）使得整个文本的汉字分布变得更加倾斜和不均，最终导致了信息熵的下降，尽管文本的总量和词汇量（独立汉字数）仍在增加。

结论：`200cn` 文本的主题专一化，导致了其汉字使用概率分布的集中。这种集中化对信息熵的拉低效应，超过了因独立汉字数增加而带来的推高效应，最终导致了熵值的净下降。

3.2 英文

对于英文，我们以“单词”（Word）为单位进行统计。

3.2.1 结果

排名	单词	次数	概率
1	and	3469	3.5039%
2	the	3262	3.2948%
3	a	2926	2.9554%
4	to	2241	2.2635%
5	of	2096	2.1171%
6	in	1527	1.5423%
7	we	1245	1.2575%
8	for	1224	1.2363%
9	that	1158	1.1696%
10	model	1148	1.1595%
11	on	912	0.9212%
12	this	818	0.8262%
13	with	798	0.8060%
14	is	675	0.6818%
15	by	618	0.6242%
16	our	469	0.4737%
17	from	444	0.4485%
18	llm	442	0.4464%
19	reasoning	426	0.4303%
20	are	418	0.4222%

图 5: 小规模样本的单词频率和信息熵

排名	单词	次数	概率
1	and	4459	3.5477%
2	the	4075	3.2422%
3	a	3648	2.9024%
4	to	2875	2.2874%
5	of	2539	2.0201%
6	in	1962	1.5610%
7	that	1544	1.2284%
8	we	1516	1.2062%
9	for	1501	1.1942%
10	model	1487	1.1831%
11	on	1126	0.8959%
12	with	1067	0.8489%
13	this	1017	0.8091%
14	is	783	0.6230%
15	by	760	0.6047%
16	llm	637	0.5068%
17	our	620	0.4933%
18	reasoning	616	0.4901%
19	from	589	0.4686%
20	task	542	0.4312%

图 6: 中规模样本的单词频率和信息熵

排名	单词	次数	概率
1	and	6863	3.4658%
2	the	6658	3.3623%
3	a	5703	2.8800%
4	to	4404	2.2240%
5	of	4142	2.0917%
6	in	3040	1.5352%
7	for	2459	1.2418%
8	we	2432	1.2282%
9	model	2372	1.1979%
10	that	2318	1.1706%
11	on	1806	0.9120%
12	this	1646	0.8312%
13	with	1633	0.8247%
14	is	1315	0.6641%
15	by	1207	0.6095%
16	our	934	0.4717%
17	from	893	0.4510%
18	method	816	0.4121%
19	data	809	0.4085%
20	an	807	0.4075%

图 7: 大规模样本的单词频率和信息熵

3.2.2 横向对比

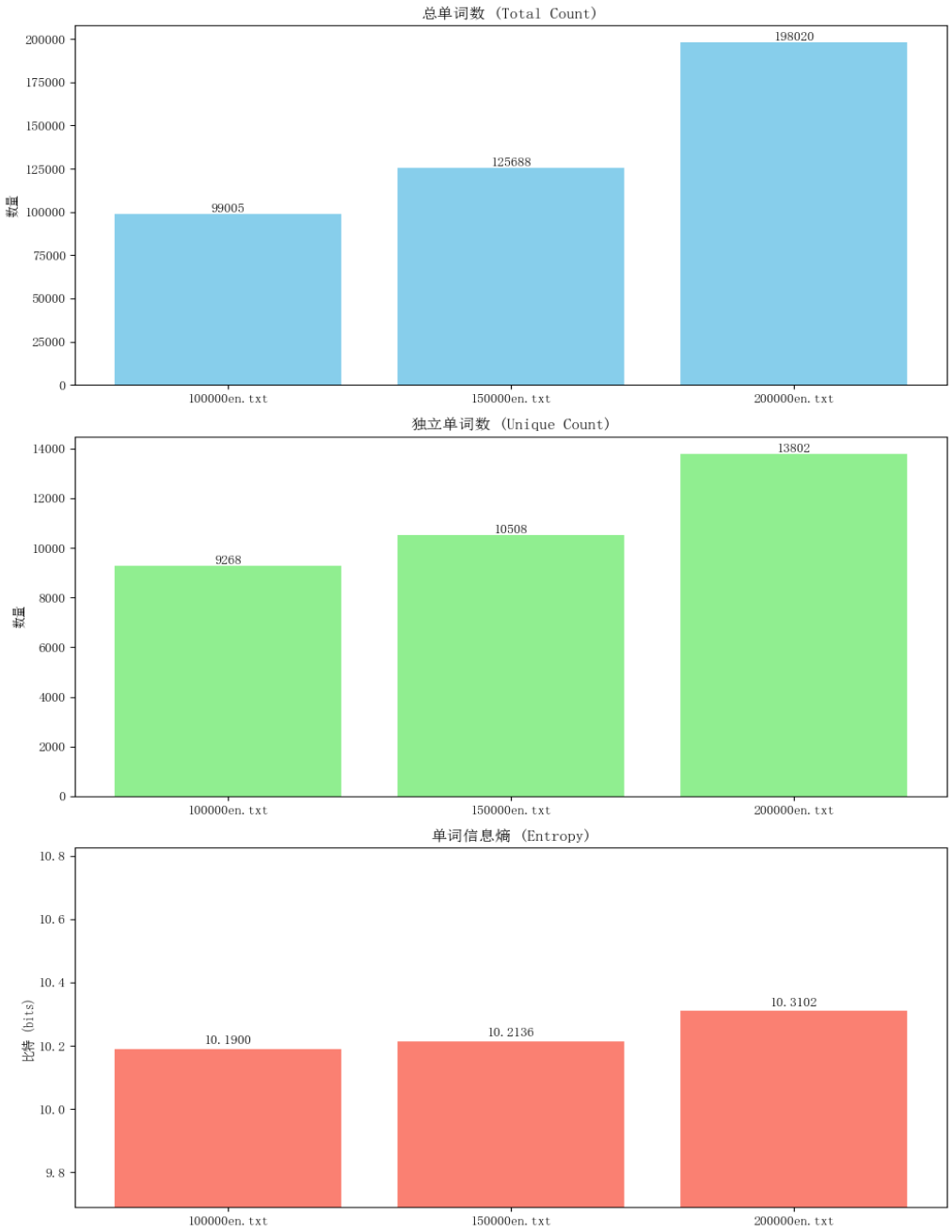

图 8: 横向对比

分析

4 验证齐夫定律 (Zipf's Law)

齐夫定律指出，在一个大型语料库中，任意单词的出现频率 f 与其在频率表中的排名 r 成反比，即：

$$f \propto \frac{1}{r}$$

取对数后，$\log(f)$ 和 $\log(r)$ 应呈线性关系。我们仅对英文样本进行单词级别的验证。

验证方法 使用 Python 实现验证逻辑：首先统计单词频率并按频率降序排列。然后计算词频与排名的对数值，通过 `numpy.polyfit` 进行线性拟合，获得斜率参数。

4.1 小规模样本

（分析...）

4.2 中规模样本

（分析...）

4.3 大规模样本

（分析...）我们在大规模样本上的对数-对数坐标图（见图 9）清晰地展示了这一线性关系，从而验证了齐夫定律。

图 1：大规模英文样本的 Zipf 定律拟合图

(`log(rank)` vs `log(frequency)`)

图 9: Zipf 定律对数-对数图