自然语言处理第一次作业（A）

作者：张宗桁
学号：2023K8009991013

2025 年 11 月 4 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：** 将所有文本转换为小写，使用 `cleaned_line.lower()`。
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 英语字母和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（字母、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 小规模样本

（此处分析小规模中文样本的汉字频率和信息熵...）

3.1.2 中规模样本

（此处分析中规模中文样本的汉字频率和信息熵...）

3.1.3 大规模样本

（此处分析大规模中文样本的汉字频率和信息熵...）

3.2 英文

对于英文，我们分别以“字母”（Letter）和“单词”（Word）为单位进行统计。

3.2.1 小规模样本

字母熵： （分析...）

单词熵： （分析...）

3.2.2 中规模样本

字母熵： （分析...）

单词熵： （分析...）

3.2.3 大规模样本

字母熵： （分析...）

单词熵： （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。

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 1）清晰地展示了这一线性关系，从而验证了齐夫定律。

图 1: Zipf 定律对数-对数图