数据

提供了两个数据集，它们是网站的形式。示例网站抓取有 10 个页面，完整的网站抓取有 500 多个页面。

示例网站爬网从两个种子链接分支出来：

- http://115.146.93.142/samplewiki/A12_scale
- http://115.146.93.142/samplewiki/Gerard_Maley

完整的网站抓取从两个种子链接分支出来：

- http://115.146.93.142/fullwiki/A12_scale
- http://115.146.93.142/fullwiki/Gerard_Maley

请注意，页面中可能存在一些爬虫陷阱和不相关的链接。您需要抓取的所有页面都将在服务器的 or 部分，其他链接可以自由忽略。数据集也可能很容易被误读，如有疑问，请使用请求库响应中的属性。
`/samplewiki//fullwiki/apparent_encodingget`

任务 1：获取所有链接（3 分）

指导背景：初步检查范围内的链接可能有助于评估流程的质量和正确性。从分析中排除的链接将是那些无法成功检索的链接，或者那些超出您评估管道的任务范围的链接（例如当前域之外的链接）。在验证您的工作时，值得记住的是，管道中较早的错误更有可能在管道后期导致更大的问题。

实现需要两个参数的函数：task1(..)task1.py

- 第一个参数是字符串列表格式的起始链接列表，并且
- 第二个参数是文件名。

您的函数应爬网每个起始链接，并使用第二个参数中给出的文件名输出格式化输出文件，并且应采用以下格式：json

{pageX： [linkXA， linkXB， ...]， pageY： [linkYA， linkYB]， ...}

其中 和 是起始链接字符串，是通过从 开始的爬网找到的页面，以及 是从 开始通过爬网找到的页面。pageXpageYlinkXAlinkXBpageXlinkYAlinkYBpageY

所有 URL 字符串都应完全限定，包括协议，例如 应该使用，即使在页面上找到的实际链接是 。每个*网页*应仅包含在每个起始链接的列表中一次，并且列表应按字母顺序排序。必须仅对与起始链接字符串位于同一域中的链接进行爬网，并且不应在输出中包含任何未成功检索的链接。

http://115.146.93.142/samplewiki/A12_scale/samplewiki/A12_scale

如果您愿意，可以自由编写任何帮助程序函数（即使在另一个文件中），但*不要更改 的函数定义*。例如，不要向 添加任何其他参数。这适用于所有任务。

task1(..)task1(..)

若要运行解决方案，请打开终端并运行 。您可以使用 根据示例数据验证您的答案，这将根据示例数据检查输出。python main.py task1 fullpython main.py

task1 sample

请注意，对于所有任务，示例数据验证并非旨在验证方法的正确性或解决方案的结构。验证输出和过程是否正确是所有任务的一部分，因此应将其视为健全性检查，确保不会错过基本格式的任何方面。在所有任务中，应将用于验证的此示例视为任意选择，因此没有隐式保证或期望它将涵盖所有情况或突出显示完整任务中存在的问题。

出于性能原因，所有任务的实现都应缓存（即下载一次并在任务期间保留在内存中）并遵守站点上的 。对于任务 2，您的代码不会在 下排除的 URL 上进行测试，但如果您愿意，当然也欢迎您在那里处理此行为。robots.txtrobots.txt

```
import pandas as pd
import json
from typing import Dict, List

import requests
import bs4
import urllib
from robots import process_robots, check_link_ok

# A simple page limit used to catch procedural errors.
SAFE_PAGE_LIMIT = 1000


# Task 1 - Get All Links (3 marks)
def task1(starting_links: List[str], json_filename: str) -> Dict[str, List[str]]:
    # Crawl each url in the starting_link list, and output
    # the links you find to a JSON file, with each starting
    # link as the key and the list of crawled links for the
    # value.
    # Implement Task 1 here

    return {}
```

任务 2：从页面中提取单词（4 分）

> 指导上下文：对于此任务，我们将查看单个页面以查看我们是否成功处理了其内容，然后再将此过程应用于后续任务中的所有页面。为了使这在诊断可能影响后续任务的问题时尽可能有用，我们将执行预处理中的所有步骤，这将包括删除与页面内容无关的常见元素（例如导航菜单和质量问题）并删除低信息词。输出旨在成为单词列表，可用于与 HTML 页面进行比较以查看已删除的内容和保留的内容。
>
> 作为警告，这种任务历来容易出错，了解您在每一步都在做什么并考虑其他方法来验证您的方法至关重要 - 您的方法的任何步骤中的任何错误都可能导致微妙的错误或重大错误，这两者都在良好的数据科学中是不可接受的。在您的方法中的某个时刻，强烈建议您花一些时间思考和尝试替代方法，并调查任何差异。

对于此任务，您应该使用 BeautifulSoup。每个页面将包含一个带有 of 的 ，处理将仅限于该处理。您可以在其中一个页面上（或通过维基百科本身）使用检查元素工具来查看这通常涵盖的内容。`dividmw-content-text`

实现一个函数，该函数采用 URL 和文件名，并输出具有该文件名的格式化文件。

JSON 对象输出的格式应为字符串形式的 URL 名称，值应为单词列表。如果无法检索页面，则列表不应包含任何值。`task2(..)task2.pyjson`

> 提取的页面应编码为其属性。`apparent_encoding`

您的函数应将其输出保存到第二个参数中给出的文件名。

预处理应分为两个阶段，第一个阶段缩小范围并从页面中删除所有不相关的元素，第二个阶段生成单词标记并删除不相关的标记。第一阶段包括八个步骤：

1. 对于其余步骤，对元素进行操作，丢弃页面的其他部分。`dividmw-content-text`
2. 从剩余的页面树中，删除带有 of 的所有元素。`thclassinfobox-label`
3. 从剩余的页面树中，删除带有 of 的所有元素。`divclassprintfooter`
4. 从剩余的页面树中，删除带有 of 的所有元素。`dividtoc`
5. 从剩余的页面树中，删除带有 of 的所有元素。`tableclassambox`
6. 从剩余的页面树中，删除带有 of 的所有元素。`divclassasbox`

7. 从剩余的页面树中，删除带有 of 的所有元素。`spanclassmw-editsection`

8. 从剩余的树中，使用 BeautifulSoup 从页面中提取文本，此提取应在元素之间使用*空格分隔符* （），**并且不应**连接来自相邻元素的文本。例如，如果剩余树内容的一部分应该变为 ，而不是 。`mw-content-text'` `'<p>a</p><p>b</p>'a b''ab'`

第一阶段的步骤 1 - 8 完成后，剩余的文本应在以下步骤中转换为标记

1. 将所有字符更改为其 ed 形式，然后将所有页面文本规范化为其形式。`casefoldNFKD`

2. 将所有非字母字符（例如，数字、撇号和标点符号）转换为单倍空格字符（例如，空格、制表符和换行符）和反斜杠字符 （） 除外。例如，应转换为 。出于此转换的目的，您应该将非英语字母字符视为非字母字符。`'\''&'' '`

3. 将所有空格字符（如制表符和换行符）转换为单空格字符，并确保每个标记之间仅存在一个空格字符。

4. 然后，应将文本转换为显式标记。此过程应使用单空格字符作为标记之间的边界。

5. 从页面文本的标记中删除 的英语停用词列表中的所有停用词。`nltk`

6. 从页面文本中删除长度小于两个字符的所有剩余标记。

7. 然后，每个令牌都应转换为其波特词干算法词干形式。这应该使用 NLTK PorterStemmer 中存在的所有默认改进（包括原始算法之外的其他改进和 NLTK 的改进）。

虽然您需要确保并验证所有步骤都已正确执行，但如果您决定在步骤 4 中使用库函数，请务必仔细阅读并完全理解文档，因为未能阅读文档可能会导致预处理步骤中的重大错误。

完成步骤 1 - 7 后，生成表示页面的 JSON 文件。JSON 文件应包含一个 JSON 对象，其中包含一个与请求的完全限定 url 匹配的字符串，关联的值应是对应于上述预处理生成的令牌的字符串数组。

如果未成功检索页面，则数组不应包含任何字符串。

若要运行解决方案，请打开终端并运行 - 这会针对示例数据的不同 URL 运行任务。您可以使用 根据示例数据验证您的答案，这将根据示例数据检查输出。对于此任务，提供了一个附加选项，通过在任务末尾添加链接，您可以查看给定 URL 的输出，例如 `.python main.py task2 fullpython main.py task2 sampleextrapython main.py task2 extra http://115.146.93.142/samplewiki/Pythagorean_tuning`

```
import json

import requests
import bs4
import urllib
import unicodedata
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

from robots import process_robots, check_link_ok


# Task 2 - Extracting Words from a Page (4 Marks)
def task2(link_to_extract: str, json_filename: str):
    # Download the link_to_extract's page, process it
    # according to the specified steps and output it to
    # a file with the specified name, where the only key
    # is the link_to_extract, and its value is the
    # list of words produced by the processing.
    # Implement Task 2 here
    return {}
```

任务 3：为所有页面制作一个单词袋（2 分）

> 指导上下文：对于此任务，我们正在考虑将前两个任务放在一起，以便从所有页面中获取每个页面上的所有单词，以生成我们所有数据的表示形式，以便以后轻松处理。这也将使我们能够在文本处理管道中查找程序错误，这些错误可能未出现在数据集的初始样本中。

实现一个函数，在该函数中，获取一个包含与任务 1 的输出格式和文件名匹配的键/值对的字典，并输出具有该文件名的格式化文件。`task3(..)task3.pycsv`

对于字典中的每个键，都应访问所有链接。应将键视为 ，将链接视为 。应根据任务 2 中的预处理的两个阶段检索每个 HTML 页面并将其处理为令牌列表，然后应将此标记列表与单个空格连接并形成与页面关联的字符串。处理完所有页面后，应创建一个数据帧，该数据帧将输出到具有给定文件名的新文件中。`seed_urllink_urllink_urlwordscsv`

输出应包含标题 、 ，每行应包含一页的详细信息，其中`csvlink_urlwordsseed_url`

- `link_url` 是单词袋对应的页面的完全限定 URL，
- `words` 是一个字符串，包含 页面的所有标记，由一个空格字符分隔，并且 `link_url`
- `seed_url` 是字典参数中链接 URL 的键。

如果在预处理后未检索到任何标记，则该字符串应为空字符串。`words''`

文件中的行应按 和 的升序排列（其中 对于两页或更多页相同）。`csvlink_urllink_urlseed_url`

要运行解决方案，请打开终端并运行 - 这将首先运行，然后使用 的输出作为输入。您可以使用 根据示例数据验证您的答案，这将根据提供的示例数据检查输出。`python main.py task3 fulltask1task1task3python main.py task3 sample`

```
from typing import Dict, List
import pandas as pd
import json
import requests
import bs4
import urllib
import unicodedata
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

from robots import process_robots, check_link_ok


# Task 3 - Producing a Bag Of Words for All Pages (2 Marks)
def task3(link_dictionary: Dict[str, List[str]], csv_filename: str):
    # link_dictionary is the output of Task 1, it is a dictionary
    # where each key is the starting link which was used as the
    # seed URL, the list of strings in each value are the links
    # crawled by the system. The output should be a csv which
    # has the link_url, the words produced by the processing and
    # the seed_url it was crawled from, this should be output to
    # the file with the name csv_filename, and should have no extra
    # numeric index.
    # Implement Task 3 here

    # Empty dataframe to demonstrate output data format.
    dataframe = pd.DataFrame()

    return dataframe
```

任务 4：绘制最常用的单词（2 分）

> 指导上下文：对于此任务，我们感兴趣的是每个种子网址的抓取中存在的常用词。我们想知道到目前为止，我们的预处理是否有效地删除了没有意义的单词，并大致了解两个词汇表中有多少重叠。

实现一个函数，在该函数中，获取与任务 3 的输出格式和一个文件名匹配的 pandas 数据帧，并将绘图输出到该文件名。 应该返回一个字典，其中每个都是一个键，值是 10 个最常见单词的列表。`task4(..)task4.pytask4(..)seed_url`

在此任务中，您应该生成一个图，该图应允许比较每个单词中的前 10 个最常见单词。这里应该使用所有页面的总出现次数的指标（例如，如果一个页面有两个单词的使用，则两者都应该计入总数）。为了输出的一致性，如果多个单词出现的频率相同，则它们应按字母顺序出现在绘图中。`seed_urlseed_url`

若要运行解决方案，请打开终端并运行 。要验证示例数据集的前 10 个单词，请使用 。`python main.py task4 fullpython main.py task4 sample`

```
import matplotlib.pyplot as plt
import pandas as pd
from typing import List, Dict
from collections import defaultdict


# Task 4 - Plotting the Most Common Words (2 Marks)
def task4(bow: pd.DataFrame, output_plot_filename: str) -> Dict[str, List[str]]:
    # The bow dataframe is the output of Task 3, it has
    # three columns, link_url, words and seed_url. The
    # output plot should show which words are most common
    # for each seed_url. The visualisation is your choice,
    # but you should make sure it makes sense for what it
    # is meant to be.
    # Implement Task 4 here

    return {}
```

任务 5：降维（3 分）

> 指导上下文：任务 4 提供了一些关于每个一般主题中经常出现的单词的一般见解。这让我们对可能想要查找的内容有了一点了解，但是如果我们想找出特定页面的主题，仅此一项可能不是最有用的工具！在这个任务中，我们感兴趣的是可视化由单词袋产生的高维空间，将其减少到两个组件进行绘图。这是验证我们的结果并确定我们的方法是否合理的另一个步骤，与所有任务一样，查看您尝试回答的问题可能会很有用。这使我们能够确定处理是否明确分离了数据，或者是否存在高度的歧义。

实现一个函数，其中获取一个与任务 3 的输出格式匹配的 pandas 数据帧和两个文件名，并将绘图输出到每个给定文件名。`task5(..)task5.py`

您应该首先生成一个单词袋，表示所有页面上的单词，然后使用 sklearn 的范数拟合到数据帧执行规范化，然后，您应该使用归一化数据帧的组件执行主成分分析。`task5(..)max2`

`task5()`应输出两个文件：

- 第一个文件的绘图输出应显示每个分量的前 10 个权重最高的标记及其权重，以及 10 个权重最高的标记及其权重。
- 输出到第二个文件的绘图应显示每个文章在 2 分量轴上的位置。来自两个数据帧的链接应包含在图中。`seed_url`

`task5()`应返回一个字典，其中 PCA 组件 ID 是键（对于解释最大方差的组件和解释下一个最大方差的组件），并且该值也是一个字典，其中此嵌套字典有四个键、，每个键的关联值如下所示：
`01"positive""negative""positive_weights""negative_weights"`

- 密钥的值应该是具有最高量级正权重的前 10 个令牌的列表。`positive`
- 密钥的值应该是具有最高量级负权重的前 10 个令牌的列表。`negative`
- 密钥的值应该是令牌列表中相关令牌的 PCA 分解中的权重。`positive_weightspositive`

- 密钥的值应该是令牌列表中相关令牌的 PCA 分解中的权重。

  `negative_weightsnegative`

如果提供了选项，则应使用 。`random_state535`

若要运行解决方案，请打开终端并运行 。要验证示例数据集的前 10 个 PCA 组件，请使用 。`python main.py task5 fullpython main.py task5 sample`

> 请注意，微小的数值误差可能会在结果中产生变异，但非数值误差也会在结果中产生变异 - 这两个因素意味着，与所有样本数据集验证一样，您不应盲目相信您的结果是正确的，而是要确保您验证了正确的过程。

任务六：分析报告（6 分）

写一份不超过 700 字的简要报告，总结你的分析结果。你应该把任务 4 和任务 5 中关于完整数据集的图画纳入你的分析中，同时包括以下内容:

比较每个任务中最常见的十个词的评论，以及为什么可能存在差异（1 分）seed_urltask4.png

解释我们在文章中发现哪些词可能不会感到惊讶，基于和（1 分）seed_urltask5a.pngtask5b.png 中的信息。

对 s 的分布的解释，以及你是否认为你可以在应用 PCA 后在二维空间绘制时确定一个新的未见过的链接的来源（1 分） urltask5b.pngseed_url

简要讨论这个数据集的局限性、处理技术的局限性以及未来可以做什么来提供进一步的见解'。(2 分)

报告应连贯、清晰、简明。 使用要点是可以接受的。(1 分)

最后这些标出来的是要传给我的文件