**大连财经学院**

**《数据采集与清洗》实训报告**

**实训名称 数据采集器高级应用实训 \_**

**专 业： \_ \_\_ \_**

**年级班级： \_ \_**

**学生姓名： \_ \_**

**学 号： \_**

**指导教师： \_ 石佳鑫 \_**

**实训日期： 2024年10月11日 \_**

**实训地点： \_ D204 \_**

**实训成绩： \_ \_**

**大连财经学院教务处制**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **实训形式** | **非独立设课实训** | √ | | **独立设课实训** | |  | **集中性综实训** | |  |
| **实训目的** | 通过实际生活中的真实案例，让学生了解数据采集的基本功能和使用方法。通过上机实训，强化学生操作方面的感性认识，明确操作的具体要求，提升动手操作能力。通过本次实训有利于学生将所学内容应用于以后的学习与工作当中，从而培养职业素养和社会责任心。 | | | | | | | | |
| **实训环境与设备** | 1.计算机一台；  2.Win10操作系统；  3.八爪鱼采集器。 | | | | | | | | |
| **实训内容** | 一、通过模板，采集凤凰网资讯即时新闻，并设置翻页次数为10。采集字段为分类、标题、来源、爬取时间、文章内容和发布时间等。    二、爬取当当网图书畅销榜Top500数据  具体内容：  1.通过目标网址http://bang.dangdang.com/books/bestsellers/01.00.00.00.00.00-recent7-0-0-1-1获取相关数据。  2.采集目标信息。最后采集的数据包括：图书名称、作者、出版社、出版时间、价格，共500条数据。    三、滚动爬取新浪微博网页  具体内容：  1.通过目标网址https://weibo.com/获取相关数据。  2.循环次数为10次。  3.采集目标信息。数据包括：用户名、发布时间和发布内容。 | | | | | | | | |
| **实训步骤** | 一、凤凰网  1. 发送请求  使用 requests 库向目标 URL 发送 HTTP 请求，获取网页 HTML 内容。  2. 解析 HTML 页面  使用 BeautifulSoup 解析网页内容，定位并提取所需字段：  标题：提取 <h1> 标签中的文本。  发布时间：查找 span 标签，class="time"。  来源：查找 span 标签，class="source"。  文章内容：查找 <div class="article-content">，提取所有 <p> 标签的内容并拼接成完整的文章正文。  分类：提取导航路径 <div class="nav"> 下的最后一个 <a> 标签文本。  3. 数据清洗与结构化  将提取到的数据整理为字典格式，便于后续存储与分析  4. 数据存储  通过 pandas 库将整理好的数据保存到 CSV 文件中，形成结构化的数据表格，方便后续分析。  二、当当网  1. 请求目标页面  使用 requests 发送 HTTP 请求，通过循环遍历前 10 页的 URL。  2. 解析页面数据  使用 BeautifulSoup 解析网页，逐本提取图书名称、作者、出版社、出版时间和价格等字段。  3. 数据存储  将提取到的数据存储为字典格式，添加到列表中，最终使用 pandas 保存为 CSV 文件。  4. 异常处理  针对可能出现的标签缺失或数据错误进行了异常处理，确保程序稳定运行。  5. 延迟请求  使用 time.sleep(1) 避免请求过快导致 IP 被封禁。  三、微博  1. 模拟浏览器滚动：  使用 Selenium 控制浏览器，模拟用户按 Page Down 键来滚动页面，加载更多微博数据。  driver.find\_element(By.TAG\_NAME, "body").send\_keys(Keys.PAGE\_DOWN) 用于模拟滚动操作。  2. 解析网页内容：  每次滚动后，获取当前页面的 HTML 内容，并使用 BeautifulSoup 解析。  提取每条微博的信息，如：用户名、发布时间、发布内容等。  3. 数据存储：  每次获取到的数据会保存在列表 data 中，最后使用 Pandas 将数据保存为 CSV 文件。  4. 延时处理：  使用 time.sleep(2) 在每次滚动后设置暂停，确保页面内容加载完成，避免数据丢失。 | | | | | | | | |
| **结果与分析** | import requests  from bs4 import BeautifulSoup  import pandas as pd  from datetime import datetime  # 目标 URL  url = "https://news.ifeng.com/c/8fO7YAofX9e"  # 定义爬取函数  def scrape\_news\_detail(url):  try:  # 发送 HTTP 请求  response = requests.get(url)  if response.status\_code != 200:  print(f"请求失败，状态码：{response.status\_code}")  return None  # 解析 HTML 页面  soup = BeautifulSoup(response.text, "html.parser")  # 提取标题  title = soup.find("h1").get\_text(strip=True)  # 提取发布时间  time\_tag = soup.find("span", class\_="index\_date\_RDRuA")  publish\_time = time\_tag.get\_text(strip=True) if time\_tag else "未知"  # 提取来源  source\_tag = soup.find("span", class\_="index\_source\_EZq0n")  source = source\_tag.get\_text(strip=True) if source\_tag else "未知"  # 提取文章内容  content\_div = soup.find("div", class\_="index\_text\_D0U1y")  if content\_div:  paragraphs = content\_div.find\_all("p")  content = "\n".join([p.get\_text(strip=True) for p in paragraphs])  else:  content = "文章内容获取失败"  # # 提取分类（假设从导航路径中获取）  # nav\_links = soup.find("div", class\_="nav").find\_all("a")  # category = nav\_links[-1].get\_text(strip=True) if nav\_links else "未知"  # 返回数据  return {  "标题": title,  # "分类": category,  "来源": source,  "发布时间": publish\_time,  "爬取时间": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),  "文章内容": content  }  except Exception as e:  print(f"爬取失败：{e}")  return None  # 主函数  def main():  # 采集单篇新闻数据  news\_data = scrape\_news\_detail(url)  if news\_data:  print("爬取成功！数据如下：")  for key, value in news\_data.items():  print(f"{key}: {value}")  # 保存为 CSV 文件  df = pd.DataFrame([news\_data])  df.to\_csv("news.csv", index=False, encoding="utf-8-sig")  print("数据已保存到 news.csv")  if \_\_name\_\_ == "\_\_main\_\_":  main()    import requests  from bs4 import BeautifulSoup  import pandas as pd  import time  # 设置请求头，模拟浏览器访问  headers = {  "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36"  }  # 目标 URL 模板  base\_url = "http://bang.dangdang.com/books/bestsellers/01.00.00.00.00.00-recent7-0-0-1-{}"  # 定义爬取函数  def scrape\_dangdang\_top500():  all\_books = [] # 存储所有图书数据  total\_pages = 10 # 爬取前 10 页数据  for page in range(1, total\_pages + 1):  print(f"正在爬取第 {page} 页数据...")  url = base\_url.format(page)  # 发送请求  response = requests.get(url, headers=headers)  if response.status\_code != 200:  print(f"请求失败，状态码：{response.status\_code}")  continue  # 解析 HTML 内容  soup = BeautifulSoup(response.text, "html.parser")  book\_items = soup.find\_all("ul", class\_="bang\_list")[0].find\_all("li")  # 遍历当前页的每一本图书  for item in book\_items:  try:  # 提取图书名称  title = item.find("div", class\_="name").find("a").get\_text(strip=True)  # 提取作者  author\_tag = item.find("div", class\_="publisher\_info").find\_all("a")  author = author\_tag[0].get\_text(strip=True) if author\_tag else "未知"  # 提取出版社  publisher = author\_tag[1].get\_text(strip=True) if len(author\_tag) > 1 else "未知"  # 提取出版时间  publish\_time = item.find("div", class\_="publisher\_info").get\_text(strip=True).split("/")[-1]  # 提取价格  price = item.find("span", class\_="price\_n").get\_text(strip=True)  # 保存到列表  all\_books.append({  "图书名称": title,  "作者": author,  "出版社": publisher,  "出版时间": publish\_time,  "价格": price  })  except Exception as e:  print(f"数据提取出错：{e}")  continue  # 避免请求过快被封禁，设置延迟  time.sleep(1)  print("数据爬取完成！")  return all\_books  # 主函数  def main():  # 爬取数据  books\_data = scrape\_dangdang\_top500()  # 保存到 CSV 文件  if books\_data:  df = pd.DataFrame(books\_data)  df.to\_csv("dangdang\_top500\_books.csv", index=False, encoding="utf-8-sig")  print("数据已保存到 dangdang\_top500\_books.csv")  else:  print("未获取到任何数据！")  if \_\_name\_\_ == "\_\_main\_\_":  main()    from selenium import webdriver  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.common.by import By  from webdriver\_manager.microsoft import EdgeChromiumDriverManager # 正确导入  from selenium.webdriver.edge.service import Service  from bs4 import BeautifulSoup  import time  import pandas as pd  # 使用 WebDriverManager 获取 Edge WebDriver  driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()))  # 访问新浪微博首页  url = "https://weibo.com/"  driver.get(url)  # 等待页面加载  time.sleep(5)  # 定义爬取函数  def scrape\_weibo():  data = []  for \_ in range(10): # 循环10次，滚动10次  # 获取页面源代码  soup = BeautifulSoup(driver.page\_source, "html.parser")  # 查找所有微博条目  weibo\_items = soup.find\_all("div", class\_="wbpro-scroller-item") # 需要根据页面实际结构进行调整  for item in weibo\_items:  try:  # 用户名  user\_name = item.find("a", class\_="ALink\_default\_2ibt1 head\_cut\_2Zcft head\_name\_24eEB").get\_text(strip=True)  # 发布时间  pub\_time = item.find("div", class\_="woo-box-flex woo-box-alignCenter woo-box-justifyCenter head-info\_info\_2AspQ").get\_text(strip=True)  # 发布内容  content = item.find("div", class\_="detail\_wbtext\_4CRf9").get\_text(strip=True)  data.append({  "用户名": user\_name,  "发布时间": pub\_time,  "发布内容": content  })  except Exception as e:  print(f"提取数据时出错：{e}")  continue  # 模拟滚动  body = driver.find\_element(By.TAG\_NAME, "body")  body.send\_keys(Keys.PAGE\_DOWN) # 向下滚动  time.sleep(2) # 每次滚动后暂停2秒，以确保加载完新内容  # 返回爬取的数据  return data  # 主函数  def main():  # 爬取数据  weibo\_data = scrape\_weibo()  # 保存数据到 CSV 文件  if weibo\_data:  df = pd.DataFrame(weibo\_data)  df.to\_csv("weibo\_data.csv", index=False, encoding="utf-8-sig")  print("数据已保存到 weibo\_data.csv")  else:  print("未获取到任何数据！")  # 关闭浏览器  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() | | | | | | | | |
| **实训体会** | 在本次实训过程中，我深入了解了如何使用 Python 进行网页数据爬取，并通过 Selenium 和 BeautifulSoup 等工具实现了自动化爬虫的构建。通过实际操作，我掌握了如何模拟用户行为，如滚动页面、点击元素等，以便从动态网页中提取数据。这次实训不仅加深了我对爬虫技术的理解，也提高了我对网页结构分析的能力。  在实训过程中，我遇到了一些问题。首先是如何正确配置 WebDriver 和浏览器驱动。在使用 Edge 浏览器时，遇到 webdriver\_manager 的版本问题，导致驱动无法正常安装或调用。通过更新相关库版本并调整驱动管理方式，我成功解决了这个问题。其次，在模拟滚动时，我发现页面加载速度较慢，导致无法及时获取到新加载的内容。为了解决这一问题，我使用了 time.sleep() 方法加入延时，确保页面内容完全加载后再进行数据提取。  通过这次实训，我不仅掌握了基本的爬虫技术，还学会了如何在遇到问题时进行排查和解决。这对我今后在数据抓取和分析方面的能力提升有很大帮助。 | | | | | | | | |
| **实训成绩** | **正确性**  **（60分）** | | **规范性**  **（20分）** | | **及时性**  **（10分）** | | | **实训纪律**  **（10分）** | |
|  | |  | |  | | |  | |
| **指导老师签字** |  | | | | | | | | |