**Python实践学期项目报告**

|  |  |
| --- | --- |
| 姓名： 樊锦东 学号： 23107410201 | |
| 科目 | Python编程实践 |
| 项目名称 | 全球天气数据储存分析与可视化  (Global Weather Data Crawling and Storage Analysis of Visualization) |

1. 项目简介
   1. 立项缘由

我在 2023-2024 学年第二学期的《数据库原理与应用》这门课上已经做出了由骆晓莉老师所评价的 “只要进一步完善就能媲美毕设”的数据库逻辑。《数据预处理》期末答辩作业选择了更难的Task2，让我初步与国际尖端文本处理的模型接触。可以说，彼时的努力给此时的我带来了课题优势。

这也是我在实践学期选题爬取天气数据的主要原因，有现成的世界范围内的地理数据预设和一些机器学习经验可供我完善和深化此实践学期的技术需求。

对我个人而言，无论是未来继续想做一个更大的项目，比如MacOS，iOS上的Objective C的程序。还是为了升学复习专业课基础知识，此次项目都提供了更多可能性的机会去探索计算机科学知识的机会。

* 1. 适用范围

对于所有在校高等教育的专科，本科，研究生，专业为数据科学与大数据技术，气象学，地质学，地理学或方向的学生及其教师和社会中的数据学家提供了连续可用的实时更新的基于Python语言的爬取数据程序。同时利用爬取数据做第二层分析，通过庞大的可获取的历史天气数据，实时对比与监控天气模型的变化并得出具有时效性的结论。

* 1. 个人背景

樊锦东，男，大连东软信息学院，人工智能技术系。（2004.05-）

百词斩全国大学生单词竞赛初赛一等奖。

普译奖第七届全国大学生写作大赛, 非英语专业组三等奖。

辽宁省第二十一届职业院校技能大赛英语写作大赛优秀奖。

普译奖第八届全国大学生翻译比赛辽宁赛区汉译英三等奖。

普译奖第八届全国大学生翻译比赛全国赛三等奖。

英伟达深度学习初级认证。

英语A级。

凯斯英语能力考试736（2023.12）。

Apple Developer Certified。

1. 项目架构与核心代码

2.1. 环境调试

2.1.1.适配项目的系统与IDE

利用macOS Sonoma 14.2, Pycharm Community Arm64 Version为基底打造的快速分布式爬虫文件。接下来的环境配置都以上面提到的设备为基准。

2.1.2.安装必要的库

在安装好Anaconda设置好环境路径后，只有一些第三方开源的库需要安装，无需多余操作。以此次项目为例，用户需要检查或安装：

1. folium
2. streamlit
3. plotly
4. scipy
5. pandas
6. numpys
7. matplotlib
8. sys
9. seaborn
10. scrapy
11. re
12. retry\_requests
13. openmeteo\_requests
14. requests\_cache

2.1.3. Anaconda环境的调配

根据Anaconda的[官方文档](https://docs.anaconda.com/anaconda/install/mac-os/)[[1]](#footnote-1)严格调试，并在Pycharm的环境编辑器里分配环境路径。具体配置图片详见图例2.1.3.1.

2.1.4.配置文件的合法性

为了避免文件与路径混乱，我对项目文件做了详细的路径规划和安排。所有的文件全在统一集成的Semester-3文件夹下，而非单独直接存在与系统根目录的多个文件夹。这样做有许多好处，使大多数人能够直接上手，具有易读性就是最具有优势的特点。使用人员也可以将文件夹作为一个插件放置到个人的总结文档中。但要注意配制爬虫中输入和输出的文件路径，如果此文件将要被放置到其他文件中去的话。具体详见[2.3.1.主体结构分析](#主体结构分析)

2.2. 项目工作流

2.2.1.项目流程解析

a. 数据获取

b. 数据收集与分析

c. 数据输出至MySQL

d. 数据可视化

2.2.2.工作流

项目在预期内获取模型或网站数据时，使用者需要先进入a.来确认是通过spider（爬虫）还是api（应用程序接口）来获取数据。

假设用户是想通过spider来获取数据，使用者需要检查网站的网址是否有效，同时确认xpath的路径是否针对网址中某一项期望要爬取的数据对应。

最终，spider在终端的运行如下:

>>> scrapy crawl qiqihar\_weather\_history -a start\_date=yyyy-mm-dd -a end\_date=yyyy-mm-dd \n  
-a output\_path=<your\_path\_here> <--nolog>

如果你想在项目下创建新的爬虫文件，详见Scrapy[官方文档](https://docs.scrapy.org/en/latest/)[[2]](#footnote-2)。

再者，欲用api（应用程序接口）来实现海量数据的离线缓存，则需要更改api接口，定制爬取地区和数据列名。

api提取数据的优点是可以专注于定义你需要的数据列，更适合那些编程基础薄弱的气象专业人员。

一旦定义好爬取文件，直接control + r 或在终端内运行:

>>> python Automatic\_Crawler\_api.py

现在我们转换到weather\_data\_collections路径下

遵循文档使用的合法性，通常来说，spider（爬虫）和api（应用程序接口）应该分别对应各自爬取的文件夹，便于之后的整理和文档微调。

文档下的路径一般情况下定义为：

>>> weather\_data\_collections/Collection\_of\_API/<Nation>/<Nation>\_weather\_daily.jsonl  
>>> weather\_data\_collections/Collection\_of\_API/<Nation>/<Nation>\_weather\_hourly.jsonl  
>>> weather\_data\_collections/Collection\_of\_Spider/<Nation>/<Nation>\_weather\_daily.jsonl  
>>> weather\_data\_collections/Collection\_of\_Spider/<Nation>/<Nation>\_weather\_hourly.jsonl

与此同时，在相同路径下也存在着一个名为Combinations的文件，利用文件里的Automatic\_Combining.py来实现洲际数据的融合，使成为一个便于后续数据对比分析或机器学习的可行文件。只需定义好输入路径和输出文件即可。

接下来就是储存至数据库和可视化。这两项完全是可选项。可以按照需求定义所有变量和第三方库。

2.3. 项目结构辩析

2.3.1.主体结构分析

接下来我们对文件中所有文件结构进行分析，方便所有人对于程序有更好的理解。

main

├── weather\_data\_acquired

│ ├── Acquired\_By\_API

│ │ ├── AutomaticScript

│ │ │ ├── .cache.sqlite

│ │ │ └── Automatic\_Crawler\_api.py

│ │ └── cn

│ │ ├── beijing

│ │ ├── dalian

│ │ ├── harbin

│ │ ├── shanghai

│ │ └── weihai (for e.g.)

│ │ ├── .cache.sqlite

│ │ ├── weihai\_weather\_daily.py

│ │ └── weihai\_weather\_hourly.py

│ └── Acquired\_By\_Spider

│ └── cn

│ └── qiqihar

│ ├── qiqihar

│ │ └── spiders

│ │ ├── \_\_init\_\_.py

│ │ ├── qiqihar\_weather\_daily.py

│ │ ├── qiqihar\_weather\_hourly.py

│ ├── \_\_init\_\_.py

│ ├── items.py

│ ├── middlewares.py

│ ├── pipelines.py

│ └── settings.py

│ └── scrapy.cfg

├── weather\_data\_collections

│ ├── Collection\_of\_API

│ │ └── <Nations> (for e.g.)

│ │ ├── <Nations>\_weather\_daily.jsonl

│ │ └── <Nations>\_weather\_hourly.jsonl

│ ├── Collection\_of\_Spider

│ │ └── cn

│ │ ├── qiqihar\_weather\_daily.jsonl

│ │ └── qiqihar\_weather\_hourly.jsonl

│ └── Combinations

│ └── EU

│ ├── eu\_daily\_weather.jsonl

│ └── eu\_hourly\_weather.jsonl

│ └── Automatic\_Combining.py

├── weather\_data\_to\_sql

│ ├── MySQLconstructor.py

│ └── tosql.png

├── weather\_data\_visualize

│ └── demonstrating

│ ├── 60deg\_latitude\_comparison\_EU.py

│ ├── Angle\_Variation.py

│ ├── Curve\_Fitting.py

│ ├── domestic-comparison.py

│ └── EU-weather-daily-analysis.py

└── output

2.3.2.天气数据存储形式与类型

1. 逐行处理

优点: 每条记录都是一行，独立于其他记录。这使得逐行处理数据变得非常简单，尤其是在流处理和实时数据处理场景中。

原因: 无需一次性加载整个文件，可以逐行读取和写入，减少内存占用，提高处理效率。

2. 数据追加

优点: 新的数据可以很方便地追加到文件末尾，而不需要重新解析和写入整个文件。

原因: 每条记录都是一行，追加操作只需在文件末尾添加新行即可。

3. 易于解析

优点: JSON 是一种轻量级的数据交换格式，易于解析和生成。许多编程语言都提供了对 JSON 的良好支持。

原因: JSON 格式的人类可读性高，数据结构清晰明了，适用于各种数据存储和传输场景。

4. 灵活性高

优点: 每行都是一个完整的 JSON 对象，可以包含不同的字段。这为存储异构数据提供了极大的灵活性。

原因: 在处理动态变化的数据结构时，JSON Lines 格式无需预定义固定的 schema，能够灵活适应不同的数据格式和内容。

5. 高效处理大数据

优点: 适合大数据处理框架，如 Hadoop、Spark 等，能够高效处理和分析大量数据。

原因: 分布式计算框架可以并行处理每一行数据，提高数据处理速度和效率。

6. 兼容性好

优点: 兼容性强，可以方便地与其他工具和系统集成，如日志管理系统、流处理平台等。

原因: JSON 是广泛使用的数据格式，许多工具和系统都支持 JSON 数据的输入和输出。

2.4. 外部连接

2.4.1.储存数据至MySQL

在此项目中，我并未选择将Scrapy框架下的pipelines管道传输直接定义至MySQL数据库中。个人认为，在如此百万级别的庞大数据库下，对于具备微调参数等特定功能的程序应该建立等级数据缓存的工作流，以适应使用者的设备性能差异。开发所用的MacBook Air在导入“EU\_hourly\_weather.jsonl”时发生了严重的卡顿，发热。“EU\_daily\_weather.jsonl”成功导入详见图例[[3]](#footnote-3)。

2.4.2.可视化至Streamlit

Streamlit 是一个开源的 Python 库，用于快速创建和分享数据应用。它简化了数据科学家、机器学习工程师和分析师的工作流程，使他们能够通过编写几行代码快速搭建互动式的数据应用和可视化工具。

简介

1. 简单易用

Streamlit 旨在让数据科学家和开发者能够在不需要前端开发经验的情况下快速构建互动式数据应用。它的 API 设计直观，只需要使用 Python 编写脚本，Streamlit 就能自动处理应用的布局和交互逻辑。

2. 开源和社区支持

Streamlit 是开源的，这意味着任何人都可以自由地使用、修改和分发。它拥有一个活跃的社区，定期发布更新和新特性，用户还可以在社区中寻求帮助和分享经验。

3. 快速迭代

Streamlit 的热重载功能使得开发者能够实时查看代码修改后的效果，这大大加快了开发和调试过程。只需保存代码文件，Streamlit 应用就会自动更新，显示最新的结果。

核心功能

1. 部署简单

Streamlit 应用可以非常方便地部署和分享。无论是在本地运行，还是部署到云服务器，Streamlit 都提供了简单的解决方案。例如，可以使用 Streamlit Cloud 或者将应用打包成 Docker 容器进行部署。

2. 互动组件

Streamlit 提供了丰富的交互式组件，如按钮、滑块、选择框、文本输入框等，用户可以通过这些组件与应用进行交互。这使得数据应用不仅仅是展示静态结果，还可以根据用户的输入动态更新和展示结果。

3. 数据可视化

Streamlit 支持多种数据可视化库，如 Matplotlib、Seaborn、Plotly 和 Altair 等。用户可以轻松地将数据可视化嵌入到 Streamlit 应用中，并提供互动功能，如图表缩放和悬停显示详细信息。

4. 布局控制

Streamlit 提供了灵活的布局控制，用户可以使用列、容器、标签页等组件来组织应用的布局。这使得应用不仅功能强大，而且界面美观，用户体验良好。

5. 文件处理

Streamlit 支持文件上传和下载功能，用户可以上传文件进行处理，或者下载处理后的结果。这在数据预处理和结果分享方面非常实用。

具体详见Streamlit[官方文档](https://docs.streamlit.io/)[[4]](#footnote-4)或[Github页面](https://github.com/streamlit/streamlit)[[5]](#footnote-5)。运用至此项目的气象数据可视化详见[2.6可视化方案](#可视化方案)和图例。

>>> pip install streamlit

>>> streamlit run <ones\_visualized\_file\_name>.py

与其他工具的对比

1. Dash 的对比

Dash 是另一个广泛使用的 Python 框架，用于构建数据应用。与 Dash 相比，Streamlit 更加注重易用性和快速迭代，代码更简洁，但 Dash 在布局和组件自定义方面提供了更强大的功能。

1. 与 Jupyter Notebook 的对比

Jupyter Notebook 是数据科学家常用的工具，用于数据分析和报告。Streamlit 与之不同，专注于构建交互式数据应用，而不仅仅是展示分析过程。Streamlit 应用更像是一个完整的 web 应用，而不是笔记本式的文档。

2.4.3.连接中转

由于在项目前数据库的E-R结构[[6]](#footnote-6)的完整性，简介性达到了很好的引导作用。我的所有暂时获取的天气数据都可以完整插入至“WeatherBroadcastSystem”中名为“Weather\_Data”的表。数据库中主键外键设计周密，具备较强的可拓展性例如数据库接入应用前端，完全可以制作一款具备时效性的天气预报软件。

2.5. 自动化脚本

2.5.1.调用天气模型

此项目中调用的是Open-Meteo网站的API。Open-Meteo 是一个强大且免费的天气数据 API 平台，通过简单易用的接口，为用户提供全球范围内的高质量天气预报数据。无论是个人开发者还是企业用户，都可以利用 Open-Meteo 提供的天气数据服务，开发出丰富多样的应用和解决方案。如果你需要获取精准的天气数据，不妨试试 Open-Meteo。

Open-Meteo 提供全球范围内的天气预报数据，包括温度、湿度、降雨量、风速和风向等多种气象参数。无论是城市还是乡村，用户都可以获取到精确的天气预报信息。用户可以通过 Open-Meteo API 获取多种时间尺度的天气数据，包括当前天气、小时预报和每日预报。这使得用户能够根据不同的需求，选择最合适的时间尺度进行数据分析和应用。Open-Meteo 提供高分辨率的天气预报数据，确保用户能够获取到精细化的气象信息。这对于需要精准天气数据的应用场景，如农业、能源和交通等领域，具有重要意义。开发者只需进行简单的 HTTP 请求，就可以获取到所需的天气数据。API 返回的数据格式为 JSON，易于解析和处理。对于个人开发者和小型项目非常友好。即使是免费的版本，也能满足大多数应用场景的需求。

2.5.2.爬取气象网站

利用Scrapy框架，构建一个主要使用xpath来爬取网页中特定位置文字等信息的工具。

>>> scrapy startproject <your\_project\_name>

>>> cd <your\_project\_name>

>>> scrapy genspider <your\_spider\_name> <website\_expected>

>>> scrapy crawl <your\_spider\_name> <output\_path> <--nolog>

2.5.3.聚集数据

接下来我们简单探讨关于一个简单的利用双重嵌套for循环的把文件中所有含有daily和hourly分别整合到一个json文件的简单程序。这个脚本的公式在我们需要处理大量文件路径中的文件时很管用。尤其是需要整合大量数据进行机器学习时。

import os  
  
# 定义根目录  
root\_dir = '/Users/fanjindong/Desktop/Semester-3/Submission/main/weather\_data\_collections/Collection\_of\_API/'  
  
# 定义输出文件路径  
daily\_output\_file = '/Users/fanjindong/Desktop/Semester-3/Submission/main/weather\_data\_collections/Combinations/EU/eu\_daily\_weather.jsonl'  
hourly\_output\_file = '/Users/fanjindong/Desktop/Semester-3/Submission/main/weather\_data\_collections/Combinations/EU/eu\_hourly\_weather.jsonl'  
  
# 函数：合并JSONL文件  
def merge\_jsonl\_files(input\_files, output\_file):  
 with open(output\_file, 'w') as outfile:  
 for input\_file in input\_files:  
 with open(input\_file, 'r') as infile:  
 for line in infile:  
 outfile.write(line)  
  
# 收集所有每日和每小时的文件路径  
daily\_files = []  
hourly\_files = []  
  
for country\_dir in os.listdir(root\_dir):  
 country\_path = os.path.join(root\_dir, country\_dir)  
 if os.path.isdir(country\_path):  
 for file\_name in os.listdir(country\_path):  
 if 'daily' in file\_name:  
 daily\_files.append(os.path.join(country\_path, file\_name))  
 elif 'hourly' in file\_name:  
 hourly\_files.append(os.path.join(country\_path, file\_name))  
  
# 合并所有每日文件  
merge\_jsonl\_files(daily\_files, daily\_output\_file)  
print(f'Merged daily files into {daily\_output\_file}')  
  
# 合并所有每小时文件  
merge\_jsonl\_files(hourly\_files, hourly\_output\_file)  
print(f'Merged hourly files into {hourly\_output\_file}')

2.6. 可视化方案

2.6.1.基础算法

1. 正逆傅立叶变换(Fourier Transform)

*Forward Fourier Transform:*

*Inverse Fourier Transform:*

*explanation:*

傅立叶变换是一种数学变换，它将一个时域信号转换到频域。它在信号处理、图像分析、物理学和工程学中有广泛应用。正傅立叶变换用于将时域信号转换为频域信号。

其中：

是频域信号（傅立叶变换结果）。

是时域信号。

是角频率。

是复指数函数。

通过傅立叶变换，我们可以将复杂的时域信号分解为多个简单的频域成分，从而更容易分析和处理。

分析

1. 频域信号的幅度

- 在三个图像中，频域信号的幅度在较低频率时较高，随着频率增加，幅度逐渐降低。这表明温度数据中存在较明显的低频成分，可能对应于季节性变化。

- 各城市的频域信号幅度变化存在差异，这反映了不同城市的气候特征和温度变化模式的不同。

2. 西北欧城市对比

- 第一幅图展示了西北欧城市（如斯德哥尔摩、奥斯陆、圣彼得堡、赫尔辛基、雷克雅未克和卑尔根）的温度数据傅立叶变换结果。

- 各城市的频域信号存在较大重叠，表明这些城市的温度变化模式具有一定的相似性。这可能是由于它们地理位置相近，气候条件相似。

3. 欧洲所有城市对比

- 第二幅图展示了更多欧洲城市的温度数据傅立叶变换结果。由于城市数量多，频域信号的重叠更加明显。

- 不同城市的频域信号在某些频率段存在差异，可能反映了不同地区的气候特点和温度变化规律。

4. 国内城市对比

- 第三幅图展示了中国国内城市（如北京、大连、上海和威海）的温度数据傅立叶变换结果。

- 国内城市的频域信号在低频段存在显著的峰值，表明这些城市的温度数据中存在较强的季节性变化。

- 各城市频域信号的幅度和形状存在差异，反映了不同城市的气候特点。

通过傅立叶变换对温度数据进行频域分析，可以揭示不同城市温度变化的频率成分和规律，为进一步的气候研究和预测提供依据。具体在程序中的应用[可见用户交互与可视化](#用户交互与可视化)。

1. 多项式回归方程(Polynomial Regression)

*equation:*

*explanation:*

通过多项式回归分析，可以有效地捕捉到各个城市在10年时间跨度内的温度变化趋势。不同城市的拟合曲线显示了各自的气候特征和温度波动模式。多项式回归提供了一种强大的工具，能够帮助我们理解和预测气候变化，尤其是在长时间尺度上的趋势和变化。这对于气候研究、城市规划和环境保护等领域具有重要意义。具体在程序中的应用[可见用户交互与可视化](#用户交互与可视化)。

2.6.2.用户交互与可视化

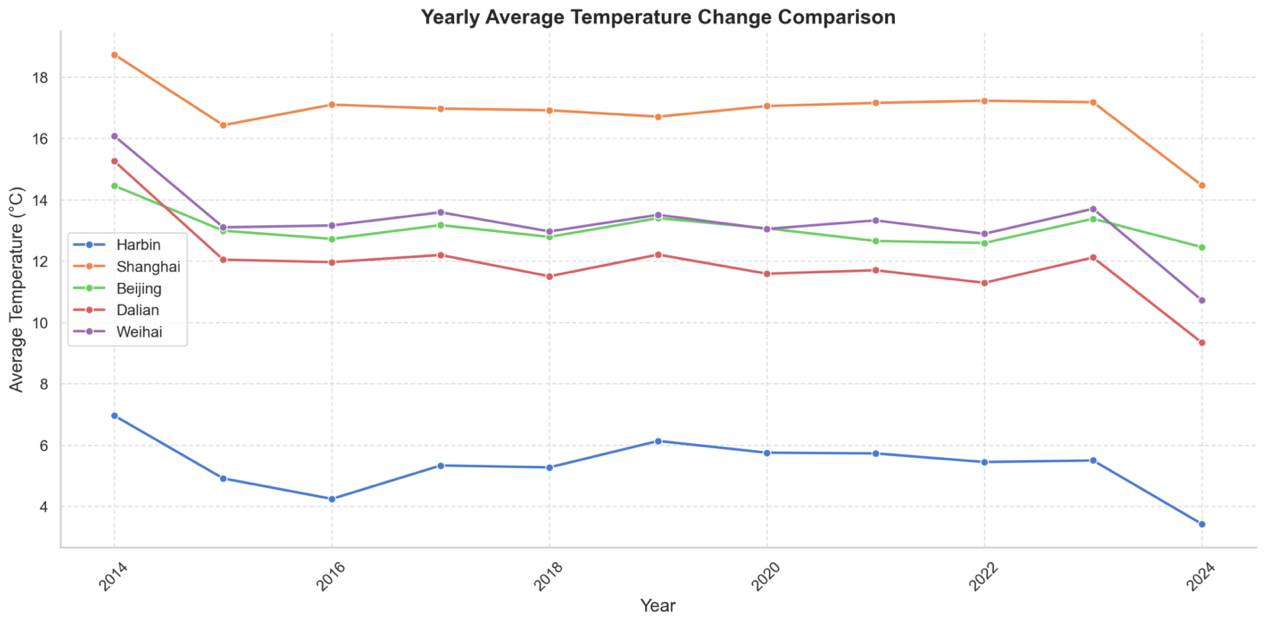
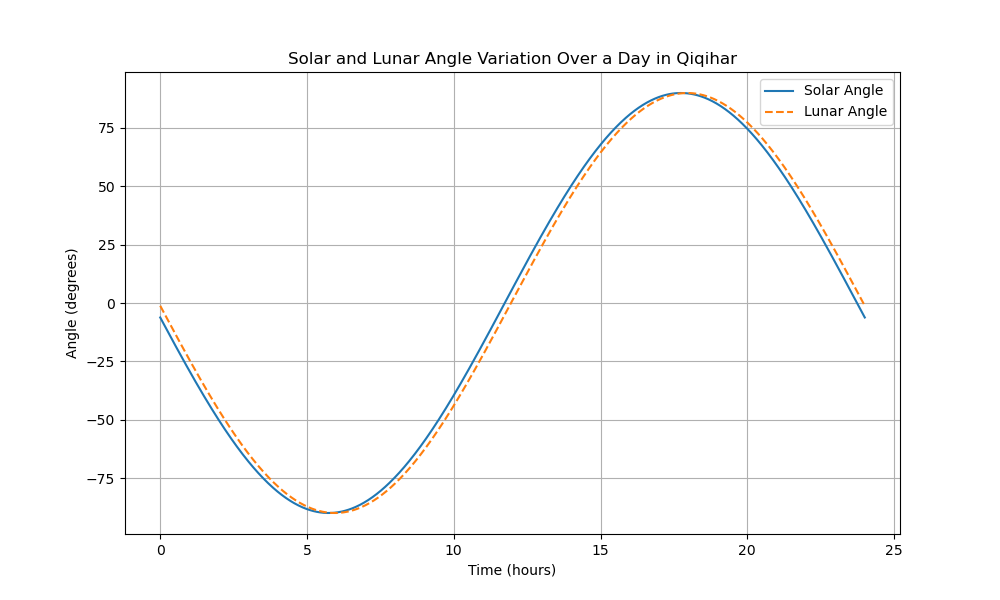
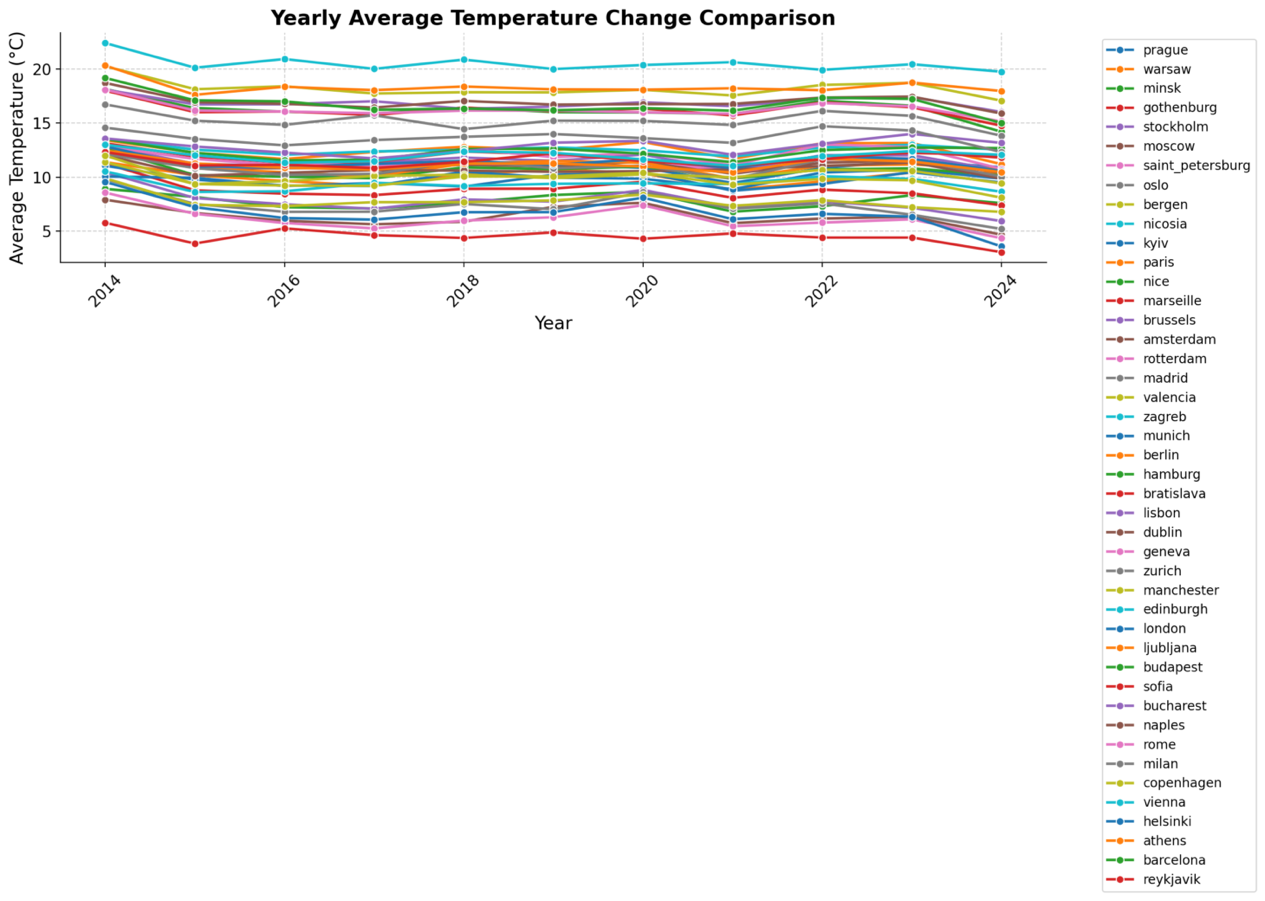
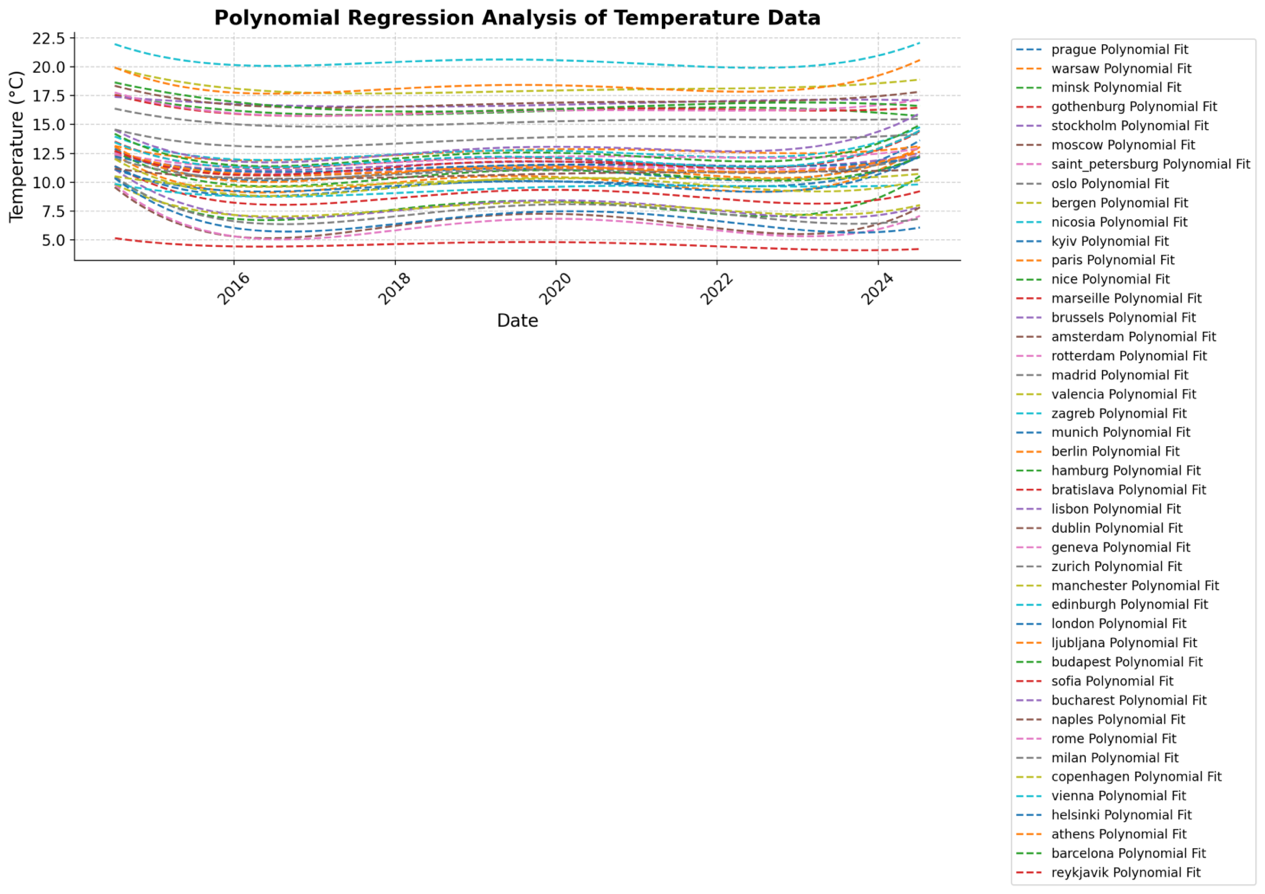
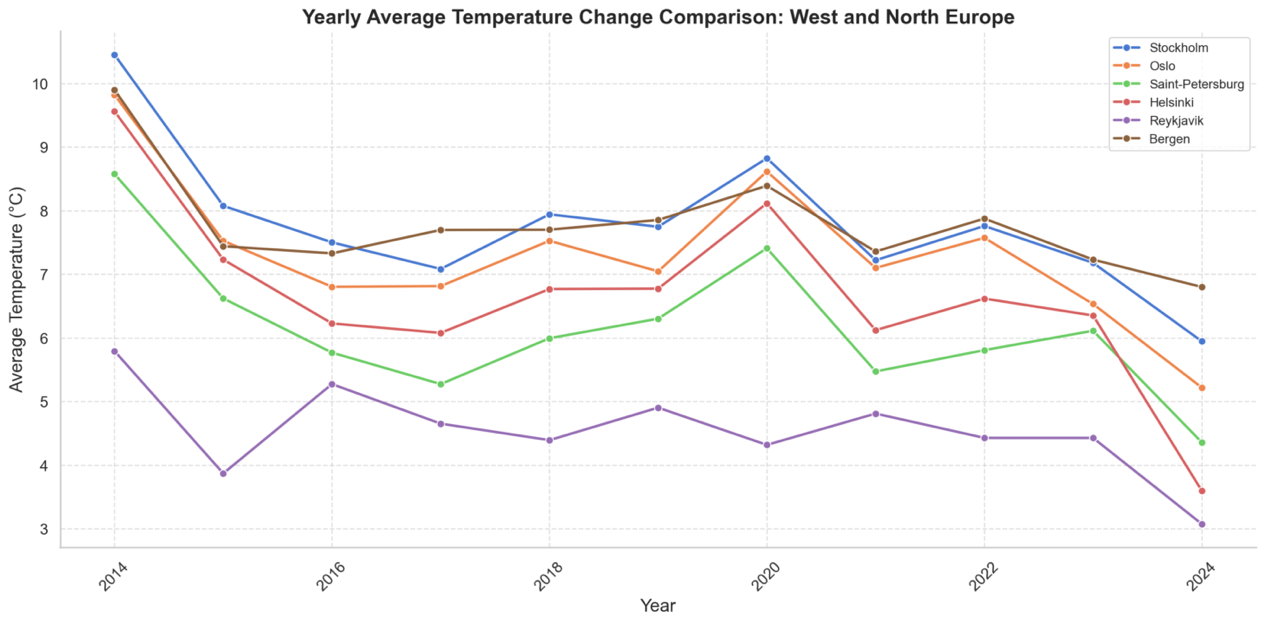
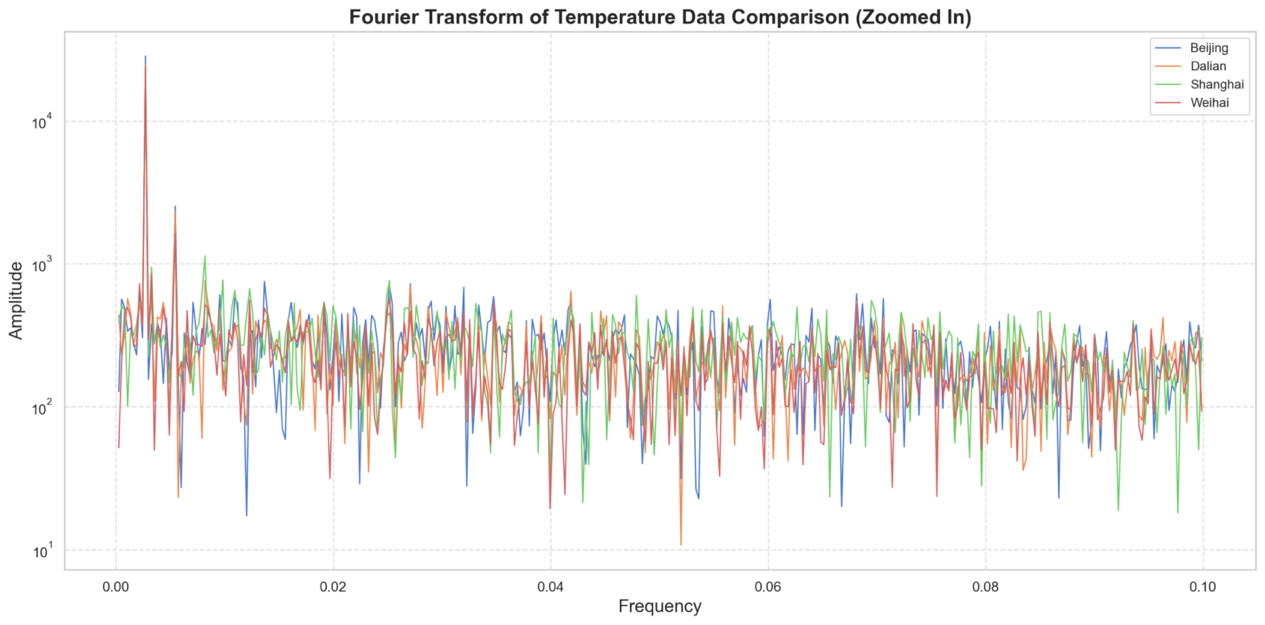
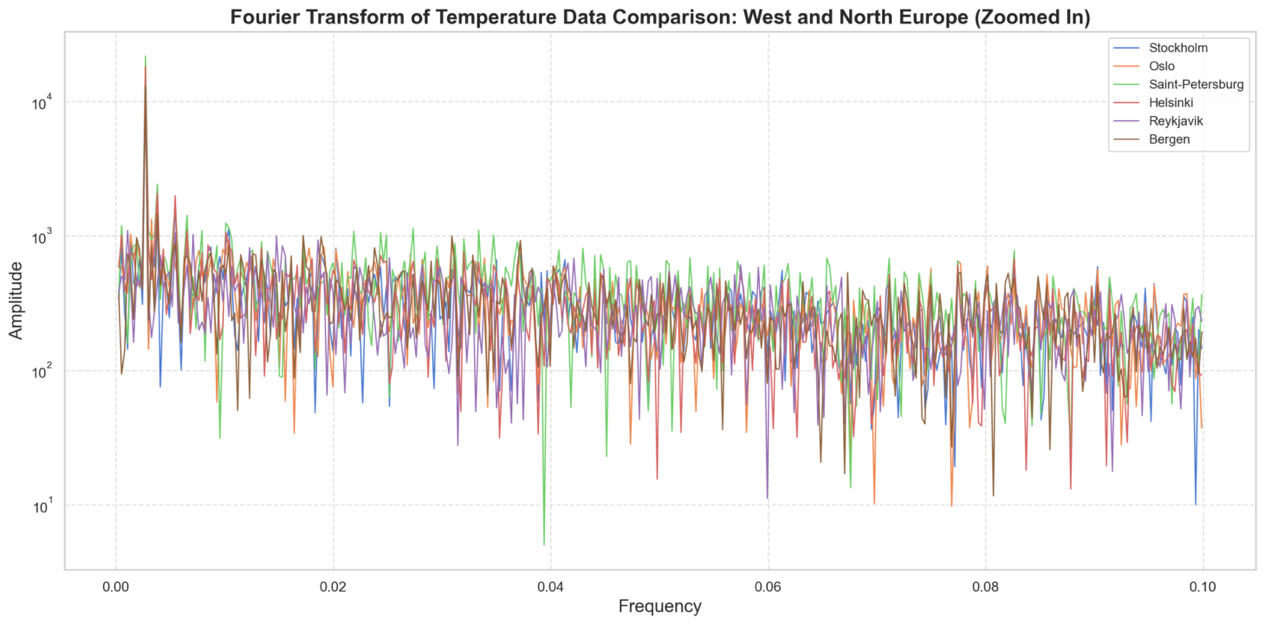
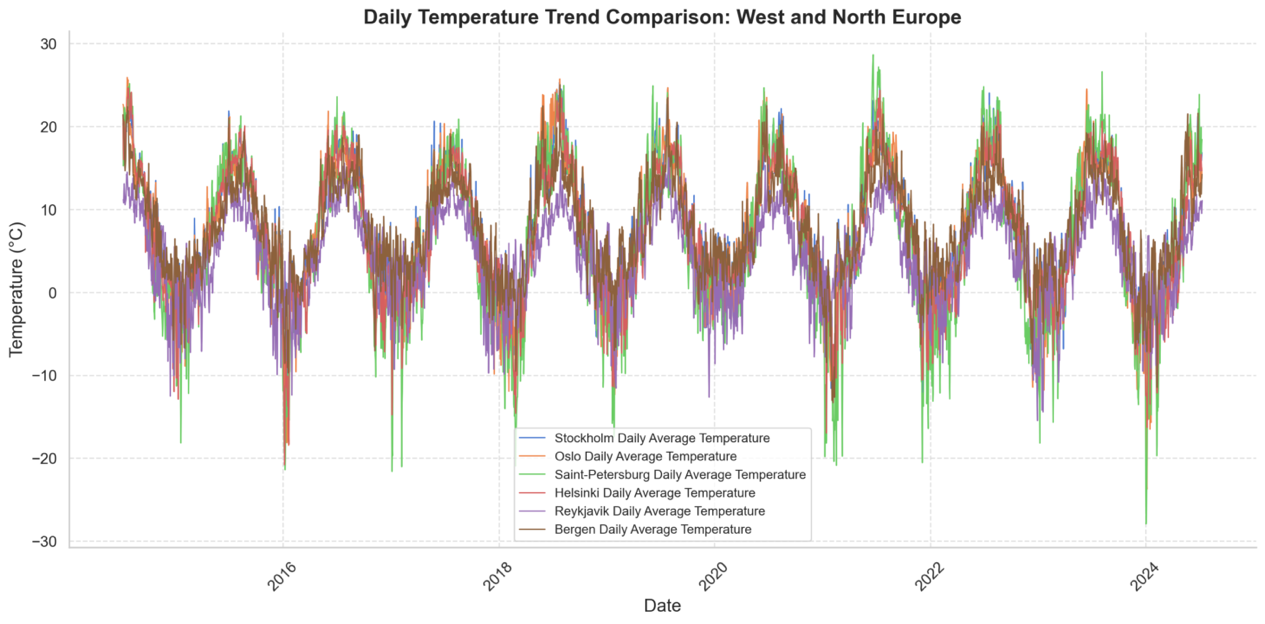
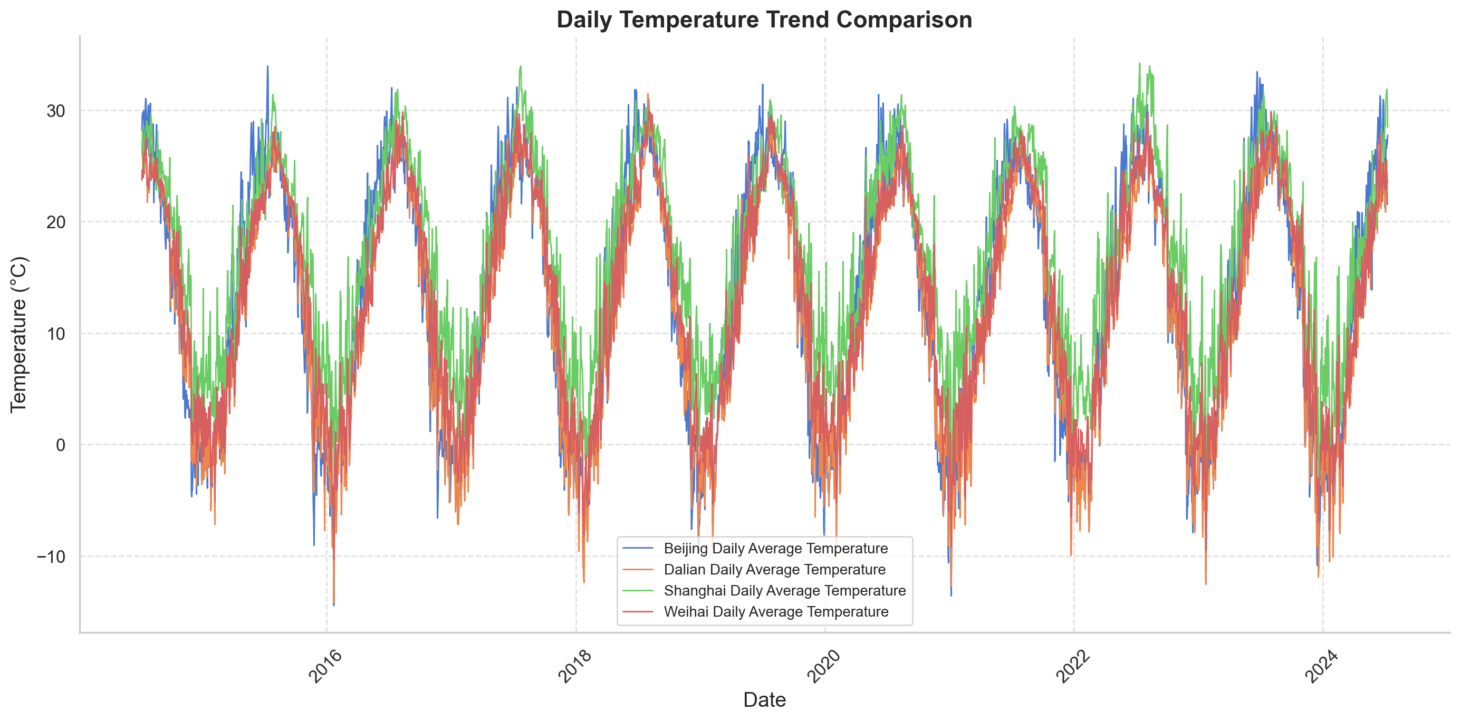
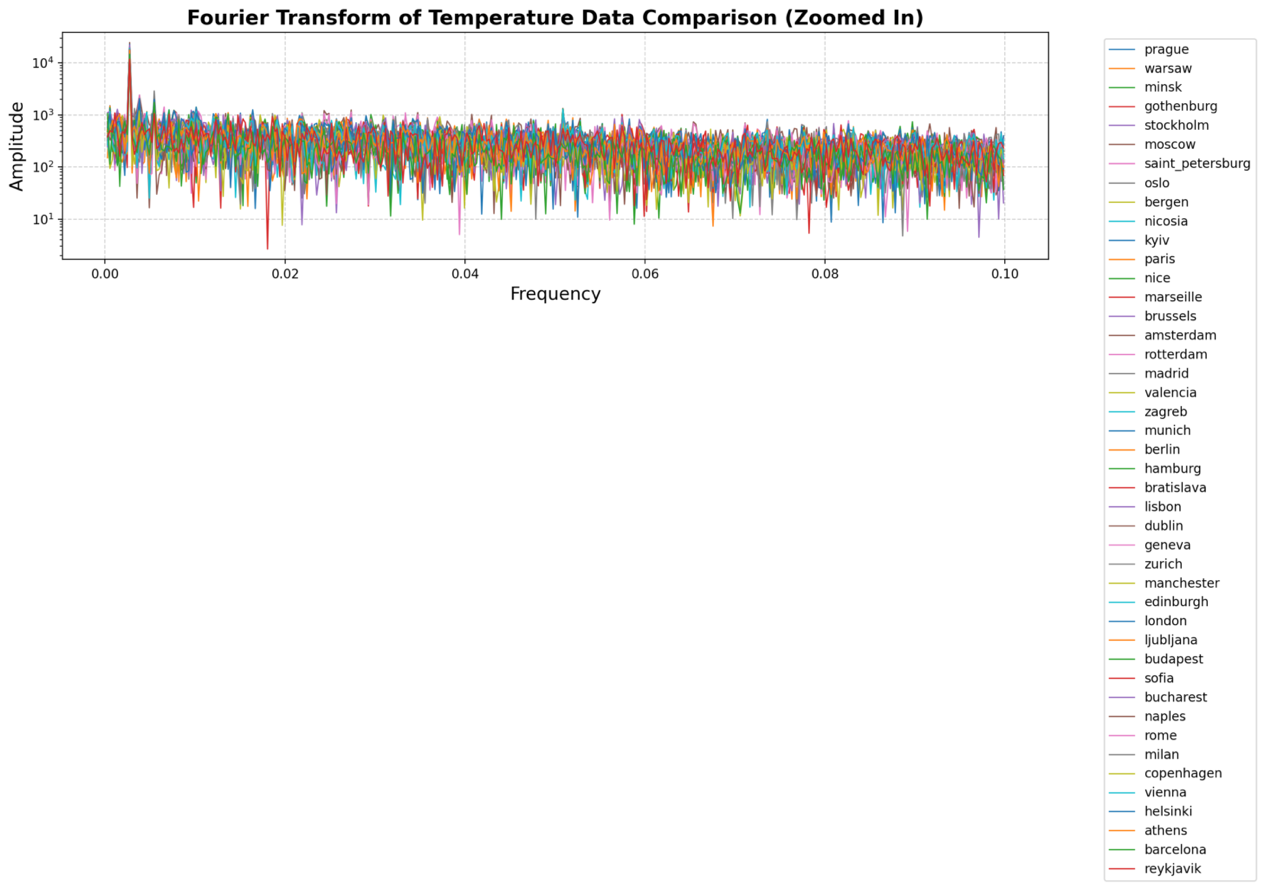
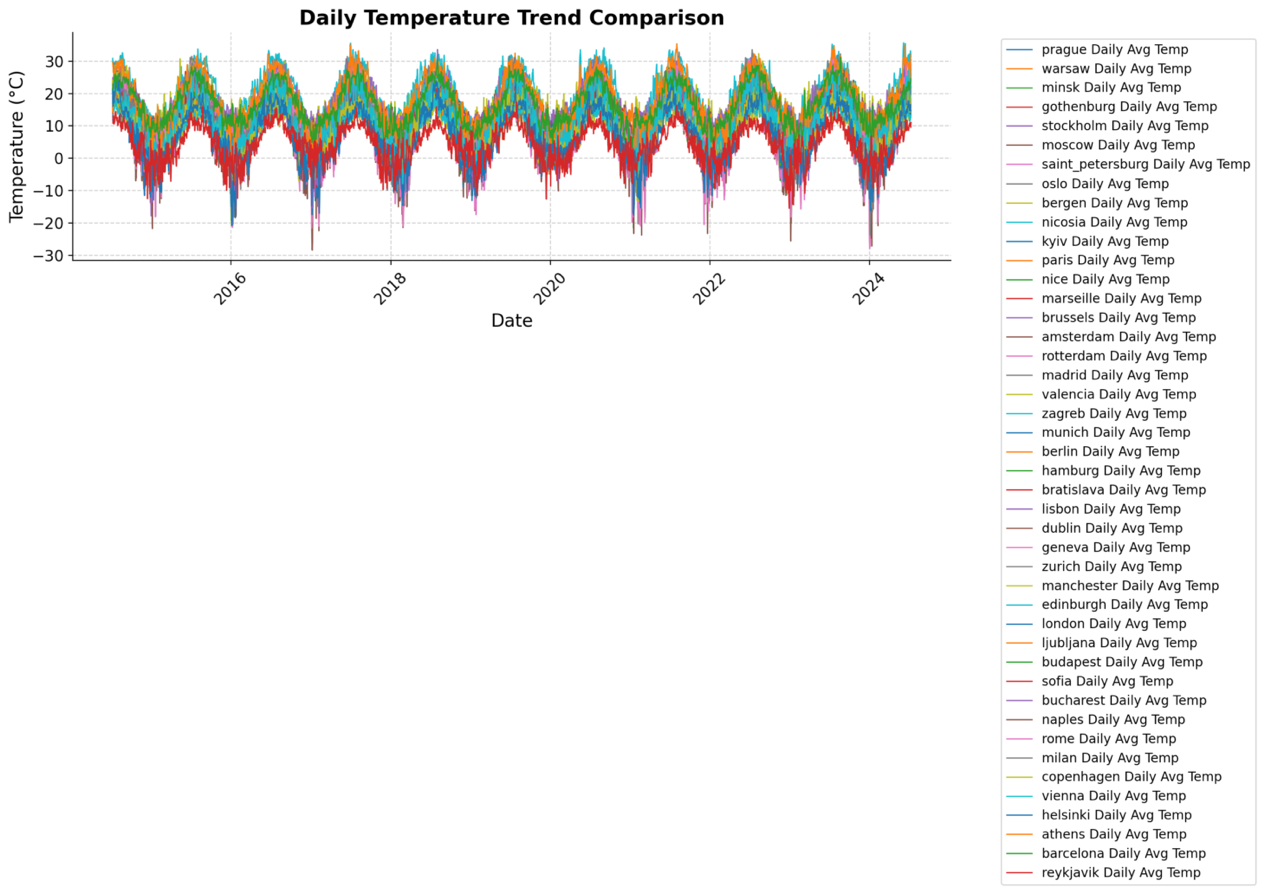
运用Streamlit中的[Openstreet](https://www.openstreetmap.org/)[[7]](#footnote-7)功能来实现用户与地区之间的交互，可以明了快速地了解一个地区的过去十年的气象信息和阳光指数。[地图交互](#地图交互)[[8]](#footnote-8)

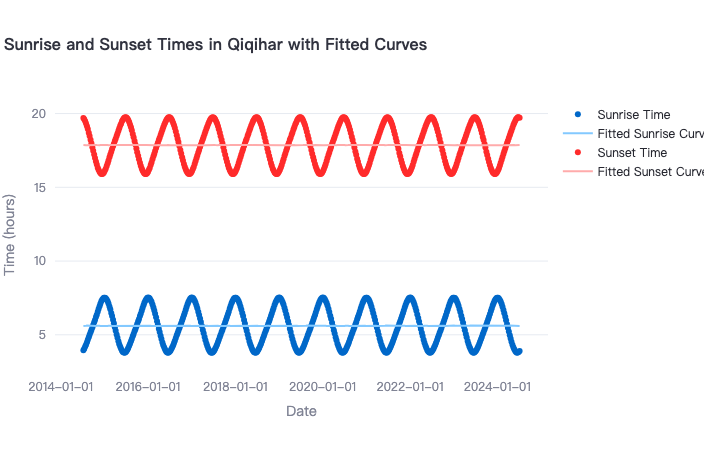
以上详见项目可视化[运行截图](#可视化文件)。

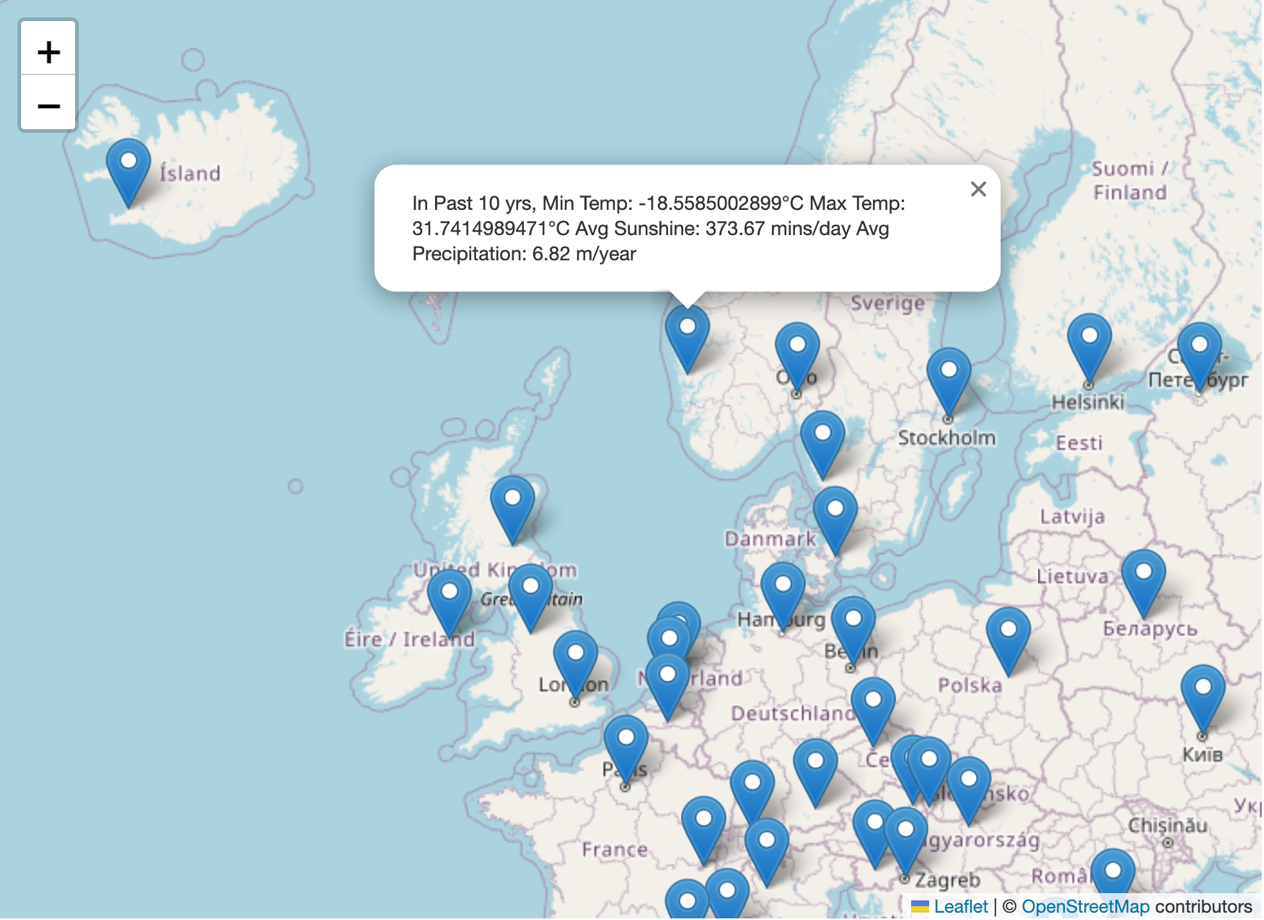
代码详见文件。

1. 项目运行截图

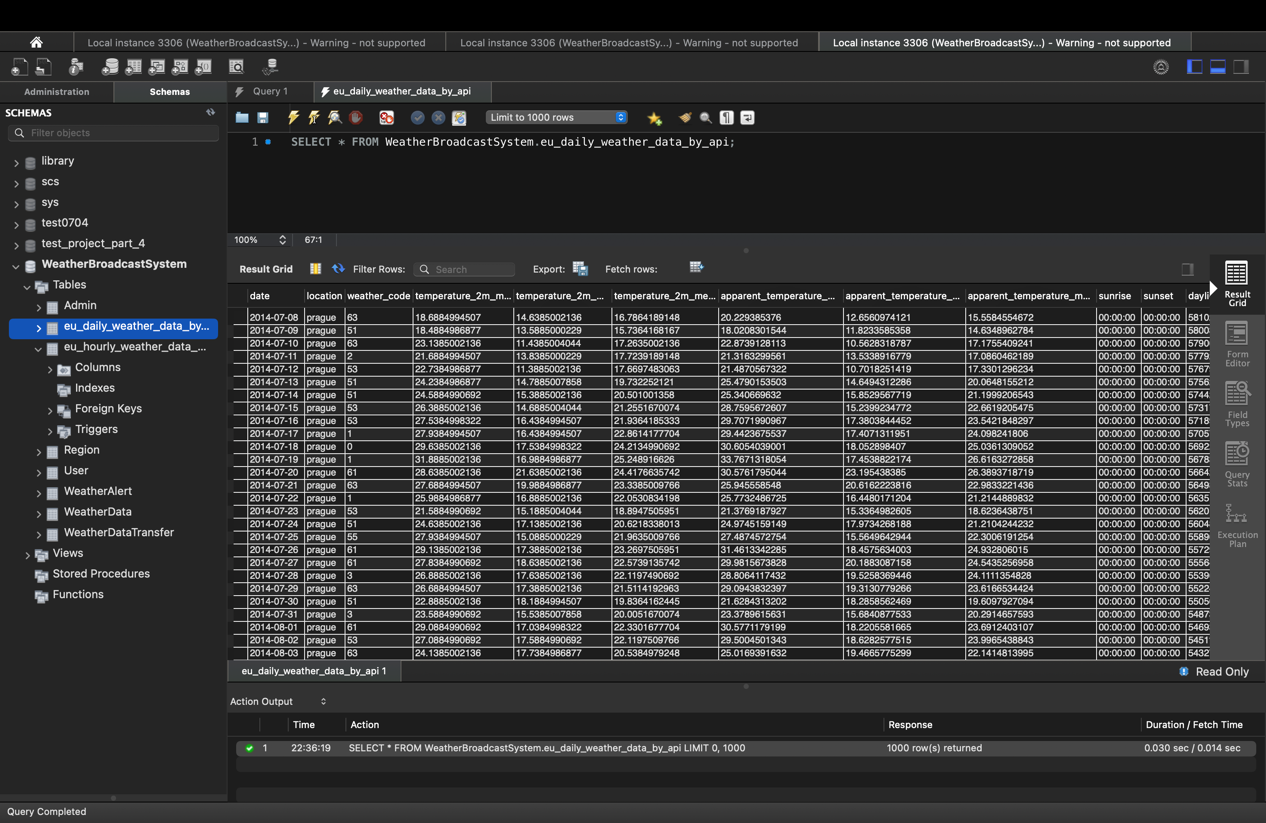
. 可视化文件

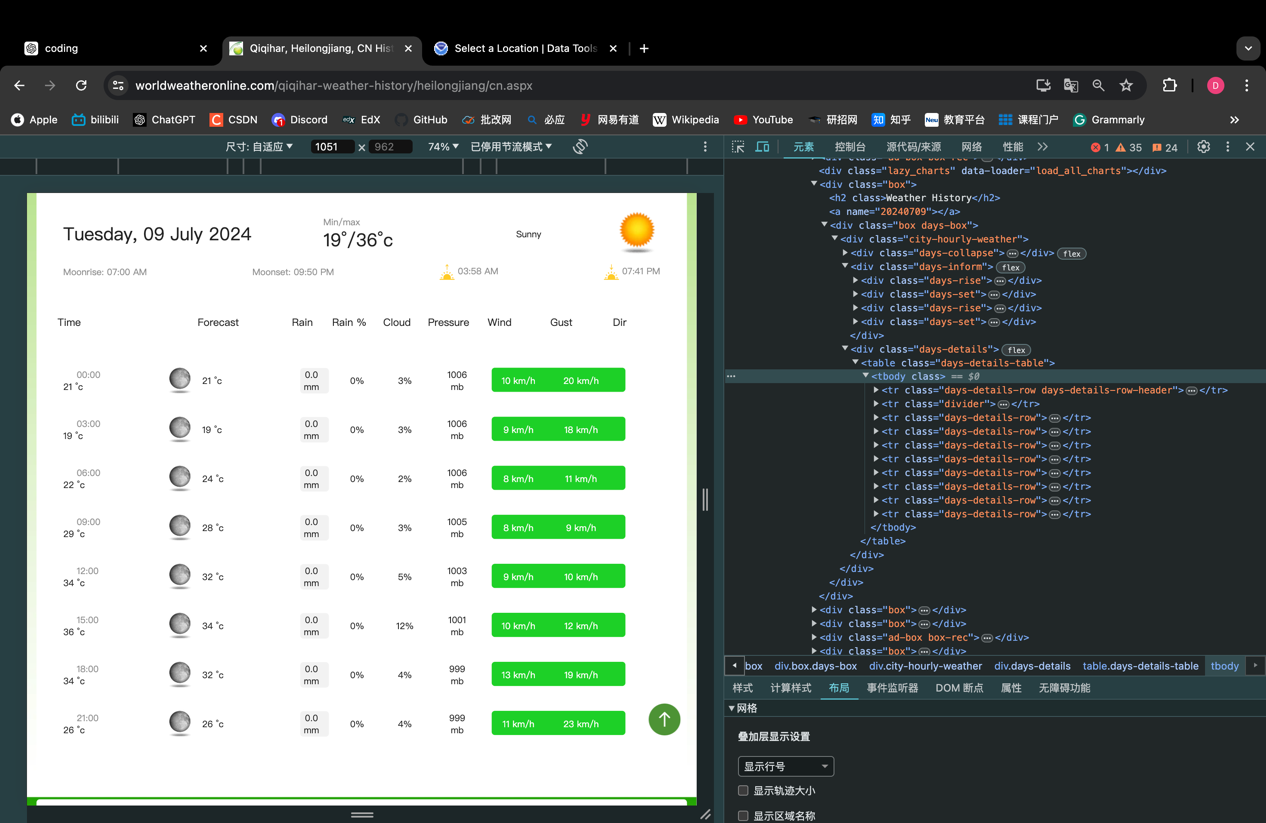
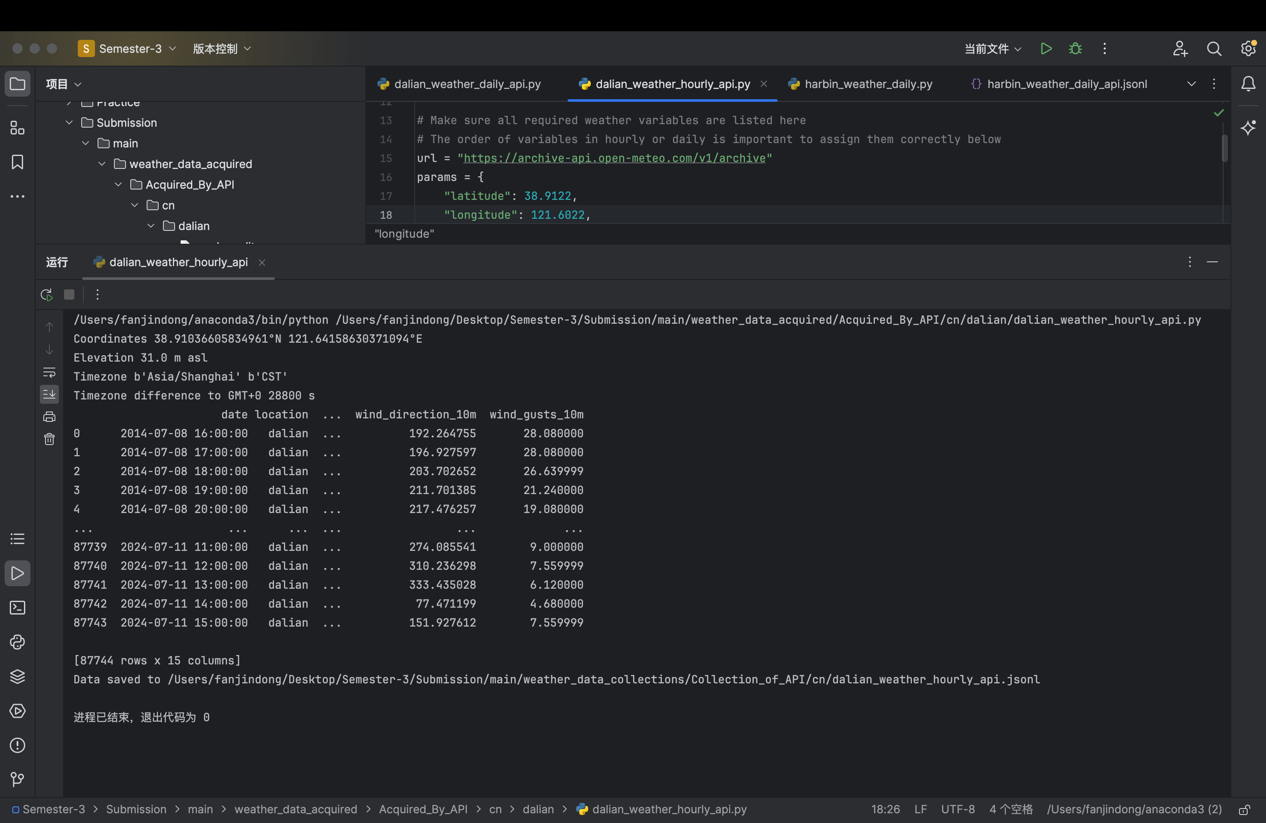






. 代码运行与网页分析





图中为爬取前仍然可用的页面，目前已失效[[9]](#footnote-9)

1. 总结

项目具有很强的继续向下深化的潜力，直到成为下一代天气预报数据模型。同时运用了两种不同的数据获取方式，一种是《数据预处理》授予Scrapy框架，撰写爬虫文件，另一种是获取免费开源的api（应用程序接口）。同时具备了算法和储存方式的代码部分，利用些许自动化脚本，使大规模储存庞大数据成为可能。

1. 改进

5.1 多样性

目前为止，项目仅获取了西部，北部欧洲，东欧，南欧部分地区和中国沿海部分城市和三座内陆城市，数据量远未达到预期。此外除了所有国家和地区外，计划在稍后的更新中推出海上洋流，风速和海面气压数据服务，有助于我们实时监测和预测未来的所有数据项。

5.2 机器学习

预计将在未来添加预测之前做出机器学习。利用显卡群运行海量气象数据，浮点计算结合算法来推测最佳预测数据。

5.3 前端UI与交互系统

我们不仅仅局限于数据可视化的交互作用。而是系统的输入地区名，或是经纬度来呈现特定地区的模型数据。上线应用市场，实现正向收入。

5.4 开源开发

预计此项目会上线至[Github](https://github.com/Sylvestervan)[[10]](#footnote-10)。

1. 参考

<https://www.youtube.com/watch?v=9P5MY_2i7K8>

<https://www.youtube.com/watch?v=cta1yCb3vA>8

<https://github.com/sammyfung/hk0weather>

https://www.youtube.com/watch?v=baqxBO4PhI8 Weather Prediction With Python And Machine Learning [W/Code] Dataquest

<https://www.ceicdata.com/zh-hans>

美国国家气象局/美国国家海洋和大气管理局（NOAA）

https://www.noaa.gov/

https://www.worldweatheronline.com/

https://en.wikipedia.org/wiki/Curve\_fitting

https://worldclim.org/data/worldclim21.html

https://open-meteo.com/en/docs/historical-weather-api

<https://docs.anaconda.com/anaconda/install/mac-os/>

https://www.worldweatheronline.com/qiqihar-weather-history/heilongjiang/cn.aspx

https://www.openstreetmap.org/

https://docs.streamlit.io/

<https://github.com/streamlit/streamlit>

澳大利亚政府气象局

﻿﻿BreezoMeter

﻿﻿中国气象局公共气象服务中心

﻿﻿德国气象局

﻿﻿加拿大环境与气候变化部

﻿﻿欧洲中期天气预报中心（ECMWE）.

﻿﻿印度气象局

﻿﻿巴西国家气象局

﻿﻿日本气象厅

和风天气

墨西哥国家气象局

泰国气象局

英国气象局

﻿﻿The Weather Channel

LaTeX

1. 致谢

厦门大学 航空航天学院 李梓旭同学

1. https://docs.anaconda.com/anaconda/install/mac-os/ [↑](#footnote-ref-1)
2. https://docs.scrapy.org/en/latest/ [↑](#footnote-ref-2)
3. 因涉及个人隐私以及其它敏感原因，代码仅展示部分导入数据。 [↑](#footnote-ref-3)
4. https://docs.streamlit.io/ [↑](#footnote-ref-4)
5. https://github.com/streamlit/streamlit [↑](#footnote-ref-5)
6. Entity-Reality Diagram详见图例 [↑](#footnote-ref-6)
7. https://www.openstreetmap.org/ [↑](#footnote-ref-7)
8. 显示为静态截图，需利用代码和数据库本地运行。 [↑](#footnote-ref-8)
9. https://www.worldweatheronline.com/qiqihar-weather-history/heilongjiang/cn.aspx [↑](#footnote-ref-9)
10. 个人主页https://github.com/Sylvestervan [↑](#footnote-ref-10)