北京地区历史天气爬虫与数据分析

姓名：魏傲宇　　学号：41621029
北京科技大学计通学院 通信 1601 班

摘要：在本报告中介绍了一套数据分析系统的需求分析、设计、实现和测试。

随着大数据、人工智能等大气探测手段的日臻完善，天气预报水平取得了显著的提升，预报准确率大幅提升。然而对于专用领域，如航空气象领域却由于成立时间晚，资料短缺，人员队伍年轻，成为掣肘航空气象预报准确率的一大因素。在此背景下，必须在引进先进设备的同时加强的历史天气的回顾总结，只有这样才能更好的服务民航发展。因此，对基于历史数据的天气预报系统的研究具有十分重要的现实意义

关键字：天气预报，爬虫，数据分析。

1. 项目背景和意义

我国横跨多个地形区，气候和地理条件均十分复杂，天气灾害分布差异巨大，且多突发性，是受天气原因影响造成自然灾害最为严重的国家之一。

随着大数据时代的来临，天气预报面临着前所未有的问题和挑战：数据的信息孤岛、系统的稳定性不足，海量数据存储和计算能力不足等都制约着气象发展，将数据挖掘技术应用于气象预报中，整合当前及历史天气形势、气象灾害影响新闻动态、数值预报产品、最新预报结论等，推论未来天气和影响，做好防灾减灾预警工作具有很大的意义。同时对内分析各种气象要素之间的内在关联，对气象预报准确率的提高也具有很大的帮助。

传统的天气预报分为两类，一是基于物理计算，结合气象各类方程，输入初始场，得出预报场，存在方程过于理想化，微尺度天气考虑不到等缺点；另一种是基于统计的概率预报，采用统计方法将历史数据要素间建立一个或多个模拟方程，往往适用于少量简单线性或者多元函数，预报对象越多，预报因子和预报对象之间的关联越难用精确的方程表示，有很大的未知性和局限性。使用相似预报技术可以针对某一设定日期，筛选大量历史数据，挖掘出与当日形势最接近的一天，找出关联，有助于提高气象预报各要素准确率，及早判断未来的影响。目前相似算法在天气预报领域的应用受到越来越多学者的关注。

目前，我国不少地市级气象部门已经把数值预报产品应用到相似预报业务中，但民航气象部门在相似预报领域的研究和应用仍然较少，在成果积累上与地方气象部门相比形成一定差距。因此，开发基于历史气象数据的天气预报系统显得尤其紧迫。[1]

2. 需求分析

天气预测对我国民航业发展有着至关重要的影响。由于我国大部分机场由于地理位置较偏，长期气候资料短缺，尤其是在复杂天气过程完全不能满足运行要求。[1]

本程序能直接爬取[天气后报网](www.tianqihoubao.com)的所有城市（此处以北京为例)历史天气信息,然后储存在 Excel 里。再将储存的信息用 Matplotlib 进行可视化、数据分析，为进一步的天气预测提供极有价值的参考信息，从而实现天气预报的自动化、智能化，给预报员更全面、直观的参考

3. 概要和详细设计

代码的总框架，各部分框图、类和函数的接口设计、界面设计、技术手段选型等。

3.1. 利用爬虫获取网站上的数据

3.1.1 函数介绍:

1).urlAnalysis:从 url 中用 request 获取 req 与 reqState，req 用于后续存储与分析 html 文件，reqState 用于判断是否继续进行操作；

2).saveHtml：从 req 中用 Beautifulsoup4 存储 html 文件

3).getHref：从 req 中取得代爬取的城市 url 列表 cityHrefList 和城市名称列表 cityNameList

4).createEmptyDir：读取文件夹，没有则创建

5). createNewExcel：读取 xlsx 文件，没有则创建

6). saveData：将数据按行列存入 xlsx 文件

7). saveDataIntoExcel：创建 xlsx 工作簿、工作表，并利用 saveData 将数据送入各行各列，如果已有数据则跳过

8). informationFilter：将爬取到的数据进行匹配和筛选

9). downloadInfo:由 cityHrefList 和 cityNameList 读取各个城市的页面，然后从各个城市的页面进入各个月份的页面；在各个页面读出改月的所有数据，再由 saveDataIntoExcel 存储至文件；另外，还要删除创建 xlsx 文件初始存在的空工作表，以便于后面的数据分析

10).main：主程序，如果 reqState=200，则开始 downloadInfo

3.1.2 函数框图:

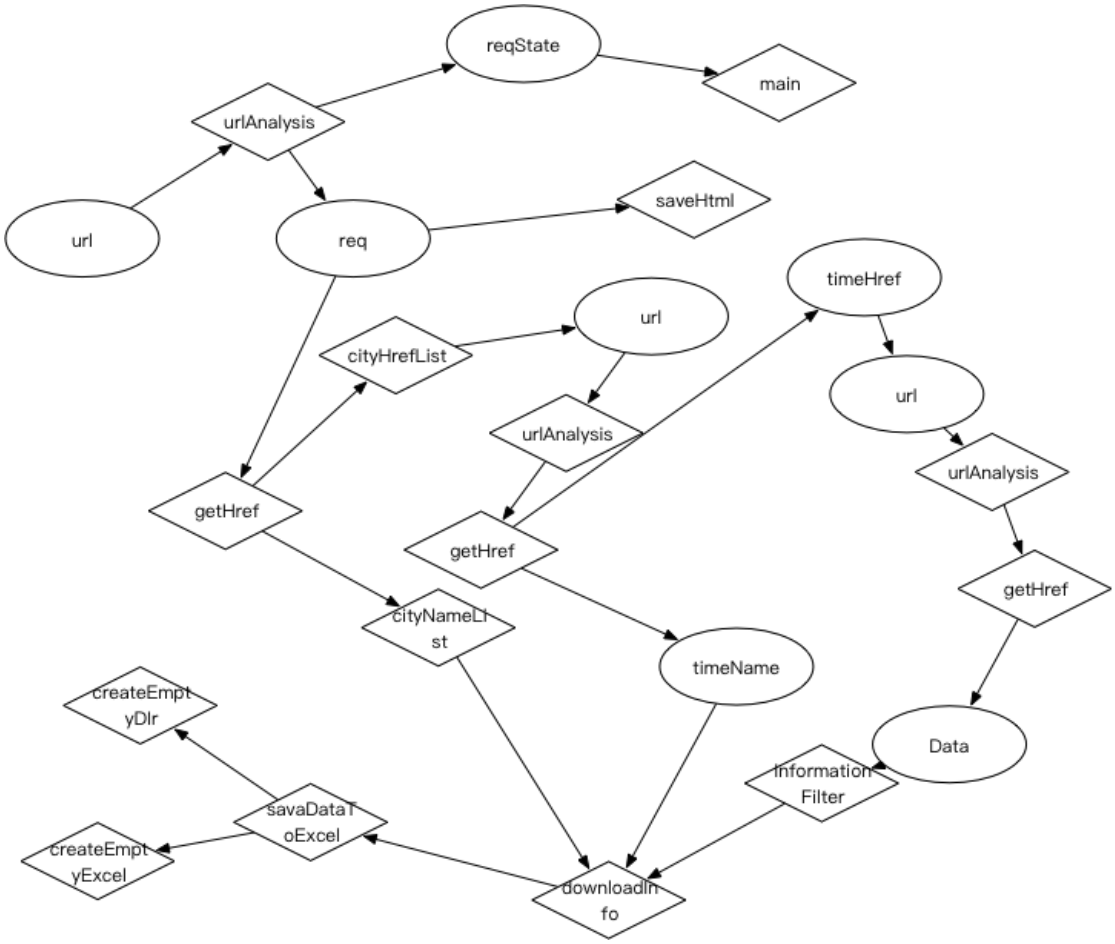

图 1 爬虫程序示意图

3.2. 将数据存储在 Excel 中

这一步已经在爬虫内完成，即一边爬取一边存储在 Excel 中

3.3. 进行自动化的可视化分析

3.3.1 函数介绍:

1).辅助函数 month_formatter:将月份数据格式化，用于后续画时间序列图操作；

2).辅助函数 hsv2rgb：hsv 数据转 rgb

3).辅助函数 color：r,g,b 数据转 rgb 字符串

4). tempGridSetting：画图操作中的坐标网格设置，包括横轴设为月份，纵轴设为温度大小

5). weatherStata：将各个气象出现情况统计在一个字典里

6). tempPlot：画图主程序，df_TAVG、df_THIG、df_TLOW、df_TD、df_AWEA、df_PWEA 分别为记录平均气温、最高气温、最低气温、每日温差、上午气象、下午气象的 DataFrame，用于画图和记录数据；首先读出 Excel 中的各个工作表各行的数据，作为每一天的数据，并存储到各个 DataFrame 里面，同时给含有温度

的 DataFrame 赋予颜色属性，颜色取决于温度的高低，由 hsv2rgb\color 求得；画出下列图表：

a）最高最低温度散点图

b）最高温度散点图，最低温度散点图，平均温度散点图，最高温差散点图，将四个图作为四个子图 subplot 拼接在一起

c）绘制气象情况条形图,统计各个气象出现天数，分 am pm

10).plt：主程序，具体的画图步骤很复杂，花了比较多的时间，但在此不多赘述

4. 代码实现

4.1 所用到的库及其版本

模块	版本
Python(os/math/time)	3.7.0
pandas	0.24.2
numpy	1.15.4
matplotlib	3.0.2
lxml	4.3.3
requests	2.19.1
beautifulsoup4	4.7.1
regex	2019.3.12
openpyxl	2.5.10

图 2 各个库及其版本声明

4.2 代码：

4.2.1 WeatherCrawler.py

```
import codecs
import requests, os
from bs4 import BeautifulSoup
from lxml import etree
import re
from openpyxl import Workbook, load_workbook
import time
```

```
# 获取 url
def urlAnalysis(url, encoding='gbk'):
    req = requests.get(url)
    req.encoding = 'gb18030'
    reqState = req.status_code
    return req, reqState


# 用 BS4 保存 Html 文件
def saveHtml(req, filename):
    soup = BeautifulSoup(req.text, features="lxml")
    with codecs.open(filename + ".html", "w", "utf-8") as f:
        f.write(soup.prettify())
    return soup


# 用 etree 得到地址和标题
def getHref(req):
    xmlContent = etree.HTML(req.text)
    cityHrefList = xmlContent.xpath('//*[@id="content"]/div[3]/dl/dd/a/@href')
    cityNameList = xmlContent.xpath('//*[@id="content"]/div[3]/dl/dd/a/text()')
    # print(cityHrefList)
    # print(cityNameList)
    return cityHrefList, cityNameList


# 创建空文件夹
def createEmptyDir(targetDir):
    if not os.path.isdir(targetDir):
        os.mkdir(targetDir)


# 创建 Excel
def createNewExcel(excelName):
    try:
        load_workbook(excelName)
    except FileNotFoundError:
        wb = Workbook()
        wb.save(excelName)


# 存储数据
def saveData(ws, monthData, dateTimeList, titleList):
    columns = 7
```

```
    rows = int(len(monthData) / columns)
    for i in range(4):
        ws.cell(row=1, column=1 + i).value = titleList[i]
    for r in range(rows):
        ws.cell(row=r + 1 + 1, column=1).value = dateTimeList[r]
        for c in range(columns):
            ws.cell(row=r + 1 + 1, column=c + 1 + 1).value = monthData[r *
columns + c]


# 保存 excel 并命名
def saveDataIntoExcel(dateTimeList, month, cityName, dateTime, yearDataList,
filename):
    titleList = ['日期', '天气状况', '气温', '风力风向']
    targetDir = os.path.join(os.path.dirname(os.path.abspath(__file__)), filename)
    createEmptyDir(targetDir)
    excelName = filename + '/' + cityName.strip() + '.xlsx'
    createNewExcel(excelName)
    wb = load_workbook(excelName)
    # process
    if dateTime not in wb.sheetnames:
        ws = wb.create_sheet(title=dateTime, index=month)
    else:
        ws = wb[dateTime]
    if ws.cell(row=3, column=3).value != None:
        print(dateTime + ' already exist')
        print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
        return
    else:
        saveData(ws, yearDataList[month], dateTimeList, titleList)
        print(dateTime + ' downloaded')
        print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
    wb.save(excelName)


# 从爬下来的数据中筛选出有用数据
def informationFilter(list):
    informationList = []
    for id in range(len(list)):
        who = re.findall('.*[\S]', list[id])
        if who != []:
            for i in range(len(who)):
                who[i] = who[i].strip()
                informationList.append(who[i])
```

```
    return informationList


# 由地址下载信息
def downloadInfo(mainurl, hrefList, nameList, maxCityNum, filename, encoding):
    if hrefList != []:
        for cityNum in range(len(hrefList)):
            while maxCityNum > 0:
                cityUrl = mainurl + hrefList[0]  # 选取对应参数可爬其他城市数据,这里只要北京
                print(mainurl + hrefList[cityNum])
                cityReq = urlAnalysis(cityUrl, encoding)[0]  # 同上
                print(nameList[cityNum])
                cityXmlContent = etree.HTML(cityReq.text)
                timeHref = cityXmlContent.xpath('//*[@id="content"]/div/ul/li/a/@href')  # 201101-201905
                timeName = cityXmlContent.xpath('//*[@id="content"]/div/ul/li/a/text()')
                print(timeHref)
                print(timeName)
                yearDataList = []
                for time in range(len(timeHref)):
                    monthNum = (timeHref[time])[-11:-5]
                    currentTimeUrl = mainurl + (f'/lishi/beijing/month/{monthNum}.html')
                    currentTimeReq = requests.get(currentTimeUrl)
                    currentTimeXmlContent = etree.HTML(currentTimeReq.text)
                    dateTimeList = currentTimeXmlContent.xpath('//*[@id="content"]/table/tr/td/a/text()')
                    dateTimeList = informationFilter(dateTimeList)
                    timeContentList = currentTimeXmlContent.xpath('//*[@id="content"]/table/tr/td/text()')
                    yearDataList.append(informationFilter(timeContentList))
                    saveDataIntoExcel(dateTimeList, month=time, cityName=nameList[cityNum], dateTime=timeName[time],
                                      yearDataList=yearDataList, filename=filename)
                print(yearDataList)
                maxCityNum -= 1
                wb = load_workbook(filename + '/' + nameList[cityNum].strip() + '.xlsx')
                wb.remove(wb["Sheet"])
                wb.save(filename + '/' + nameList[cityNum].strip() + '.xlsx')
```

```
    else:
        print('Failed')


# 主程序
def main(mainurl, reqState, req, maxCityNum, filename, encoding):
    if reqState == 200:
        print('Url Request Accepted')
        # saveHtml(req, filename)
        hrefList, nameList = getHref(req)
        downloadInfo(mainurl, hrefList, nameList, maxCityNum, filename,
encoding)
    else:
        print('Url Request Denied')


'''
mainurl = 'http://www.tianqihoubao.com'
url = 'http://www.tianqihoubao.com/lishi'
encoding = 'gbk'
head = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)'
                      ' AppleWebKit/605.1.15 (KHTML, like Gecko)'
                      ' Version/12.0.1 Safari/605.1.15',
        'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',}
filename = 'WeatherHistory'
req, reqState = urlAnalysis(url,filename)
maxSetNumber = 1

if reqState == 200:
    main(mainurl, reqState, req, maxSetNumber, filename,encoding)
else:
    print(reqState)

'''
```

4.2.2 DataAnlysis.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator, FuncFormatter
from pandas import DataFrame
import math, os
```

```
from matplotlib.font_manager import FontProperties

font_set = FontProperties(fname='PingFang', size=12)


# 月份坐标轴
def month_formatter(x, pos):
    v = int(x / 31) + 1
    if v > 12:
        return '-'
    else:
        return str(v)


# hsv 转 rgb
def hsv2rgb(h, s, v):
    h = float(h)
    s = float(s)
    v = float(v)
    h60 = h / 60.0
    h60f = math.floor(h60)
    hi = int(h60f) % 6
    f = h60 - h60f
    p = v * (1 - s)
    q = v * (1 - f * s)
    t = v * (1 - (1 - f) * s)
    r, g, b = 0, 0, 0
    if hi == 0:
        r, g, b = v, t, p
    elif hi == 1:
        r, g, b = q, v, p
    elif hi == 2:
        r, g, b = p, v, t
    elif hi == 3:
        r, g, b = p, q, v
    elif hi == 4:
        r, g, b = t, p, v
    elif hi == 5:
        r, g, b = v, p, q
    r, g, b = int(r * 255), int(g * 255), int(b * 255)
    return r, g, b


# rgb 转 16#
```

```
def color(value):
    digit = list(map(str, range(10))) + list("ABCDEF")
    if isinstance(value, tuple):
        string = '#'
        for i in value:
            a1 = i // 16
            a2 = i % 16
            string += digit[a1] + digit[a2]
        return string


def tempGridSetting():
    plt.gca().xaxis.set_major_locator(MultipleLocator(31))
    plt.gca().xaxis.set_major_formatter(FuncFormatter(month_formatter))
    plt.setp(plt.gca().get_xticklabels(), horizontalalignment='center')
    plt.gca().yaxis.set_major_locator(plt.MultipleLocator(10))
    plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, y: str(x) + 'C°'))
    plt.gca().grid()


def weatherStata(WeatherList):
    weatherDic = {}
    for weather in WeatherList:
        if weather in weatherDic.keys():
            weatherDic[weather] += 1
        else:
            weatherDic[weather] = 1
    return weatherDic


def tempPlot(xlsxFile):
    # 初始化
    avgTemp = []
    lowestTemperature = []
    highestTemperature = []
    dateTime = []
    amWeatherList = []
    pmWeatherList = []
    windStrenthList = []
    sheetNameList = xlsxFile.sheet_names

    df_TAVG = DataFrame(columns=['datetime', 'temp', 'color'])  # 平均气温
数据
    df_THIG = DataFrame(columns=['datetime', 'temp', 'color'])  # 最高气温数
```

```
据
        df_TLOW = DataFrame(columns=['datetime', 'temp', 'color'])  # 最低气温
数据
        df_TD = DataFrame(columns=['datetime', 'temp', 'color'])  # 最低气温数据
        df_AWEA = DataFrame(columns=['datetime', 'weather'])  # 上午气象数据
        df_PWEA = DataFrame(columns=['datetime', 'weather'])  # 下午气象数据

        print(len(sheetNameList))

        # 存入各个数据及其对应日期
        for sheetName in range(len(sheetNameList)):
            currentSheet = xlsxFile.parse(sheetNameList[sheetName])
            for row in range(currentSheet.iloc[:, 0].size):
                dateTime.append(currentSheet.iloc[[row]].values[0][0])

                amWeatherList.append(currentSheet.iloc[[row]].values[0][1])
                pmWeatherList.append((currentSheet.iloc[[row]].values[0][2])[1:])


highestTemperature.append(int((currentSheet.iloc[[row]].values[0][3])[:-1]))

lowestTemperature.append(int((currentSheet.iloc[[row]].values[0][5])[:-1]))

                windStrenthList.append((currentSheet.iloc[[row]].values[0][6])[-4:-1])

        # 计算平均气温 and 将相关数据存入 DataFrame
        for i in range(len(lowestTemperature)):
            aT = (lowestTemperature[i] + highestTemperature[i]) / 2
            avgTemp.append(aT)

            # 平均气温 TAVG
            _color = color(tuple(hsv2rgb(int(-7 * aT) + 240, 1, 1)))  # 设置数据颜色
值
            _df = DataFrame({'datetime': [dateTime[i]], 'dayIndex': [(i + 1) % 365],
                             'temp': [aT], 'color': [_color]})
            df_TAVG = df_TAVG.append(_df, ignore_index=True)

            # 最高气温 HAVG
            _color = color(tuple(hsv2rgb(int(-7 * highestTemperature[i]) + 240, 1, 1)))
            _df = DataFrame({'datetime': [dateTime[i]], 'dayIndex': [(i + 1) % 365],
                             'temp': [highestTemperature[i]], 'color': [_color]})
            df_THIG = df_THIG.append(_df, ignore_index=True)

            # 最低气温 LAVG
```

```
            _color = color(tuple(hsv2rgb(int(-7 * lowestTemperature[i]) + 240, 1, 1)))
            _df = DataFrame({'datetime': [dateTime[i]], 'dayIndex': [(i + 1) % 365],
                             'temp': [lowestTemperature[i]], 'color': [_color]})
            df_TLOW = df_TLOW.append(_df, ignore_index=True)

            # 温差 TD
            _color = color(tuple(hsv2rgb(int(-7 * (highestTemperature[i] -
lowestTemperature[i])) + 240, 1, 1)))
            _df = DataFrame({'datetime': [dateTime[i]], 'dayIndex': [i % 365],
                             'temp': [highestTemperature[i] -
lowestTemperature[i]], 'color': [_color]})
            df_TD = df_TD.append(_df, ignore_index=True)

            # 上午天气 AWEA
            _df = DataFrame({'datetime': [dateTime[i]], 'dayIndex': [(i + 1) % 365],
                             'weather': [amWeatherList[i]]})
            df_AWEA = df_AWEA.append(_df, ignore_index=True)

            # 下午天气 PWEA
            _df = DataFrame({'datetime': [dateTime[i]], 'dayIndex': [(i + 1) % 365],
                             'weather': [pmWeatherList[i]]})
            df_PWEA = df_PWEA.append(_df, ignore_index=True)

    df_TAVG = df_TAVG.set_index('datetime')
    df_THIG = df_THIG.set_index('datetime')
    df_TLOW = df_TLOW.set_index('datetime')
    df_TD = df_TD.set_index('datetime')

    #
    if not os.path.isdir('./pic'):
        os.mkdir('./pic')
    # 最高最低温度散点图
    tempGridSetting()
    plt.figure(figsize=(12, 9))
    plt.scatter(df_TLOW['dayIndex'], df_TLOW['temp'],
                c='mediumpurple', marker='o', alpha=0.05)
    plt.scatter(df_THIG['dayIndex'], df_THIG['temp'],
                c='red', marker='o', alpha=0.05)
    plt.ylim(-30, 50)
    plt.title('TempCurve : 2011-2019')
    plt.xlabel("Month")
    plt.ylabel("Temperature C°")
    plt.legend()
    plt.savefig("./pic/HighLowTemp.png", bbox_inches='tight')   # 保存图片
```

```
plt.show()

# 绘制平均温度散点图
plt.figure(figsize=(12, 9))
plt.subplot(2, 2, 1)
tempGridSetting()
plt.scatter(df_TAVG['dayIndex'], df_TAVG['temp'], s=80,
            c=df_TAVG['color'], marker='o', alpha=0.05)
plt.ylim(-30, 50)
plt.title('avgTemp : 2011-2019')
plt.xlabel("Month")
plt.ylabel("Temperature C°")

# 绘制最高温度散点图
plt.subplot(2, 2, 2)
tempGridSetting()
plt.scatter(df_THIG['dayIndex'], df_THIG['temp'], s=80,
            c=df_THIG['color'], marker='o', alpha=0.05)
plt.ylim(-30, 50)
plt.title('higTemp : 2011-2019')
plt.xlabel("Month")
plt.ylabel("Temperature C°")

# 绘制最低温度散点图
plt.subplot(2, 2, 3)
tempGridSetting()
plt.scatter(df_TLOW['dayIndex'], df_TLOW['temp'], s=80,
            c=df_TLOW['color'], marker='o', alpha=0.05)
plt.ylim(-30, 50)
plt.title('lowTemp : 2011-2019')
plt.xlabel("Month")
plt.ylabel("Temperature C°")

# 绘制温度差异散点图
plt.subplot(2, 2, 4)
tempGridSetting()
plt.scatter(df_TD['dayIndex'], df_TD['temp'], s=80,
            c=df_TD['color'], marker='o', alpha=0.05)
plt.ylim(-30, 50)
plt.title('TD : 2011-2019')
plt.xlabel("Month")
plt.ylabel("Temperature C°")
plt.savefig("./pic/Temp.png", bbox_inches='tight')   # 保存图片
plt.show()
```

```
# 绘制气象情况条形图
amWeatherDic = weatherStata(df_AWEA['weather'])
pmWeatherDic = weatherStata(df_PWEA['weather'])
aList = sorted(amWeatherDic, key=amWeatherDic.__getitem__, reverse=True)
pList = sorted(pmWeatherDic, key=pmWeatherDic.__getitem__, reverse=True)
aValue = []
pValue = []
for key in aList:
    aValue.append(amWeatherDic[key])
for key in pList:
    pValue.append(pmWeatherDic[key])

plt.figure(figsize=(12, 9))
plt.subplot(2, 1, 1)
font_set = FontProperties(fname="simsun.ttc", size=12)
plt.setp(plt.gca().get_xticklabels(), horizontalalignment='center')
plt.xticks(range(len(aList)), aList, FontProperties=font_set, rotation=60)
plt.bar(aList, aValue)
plt.ylim(0, 1300)

plt.subplot(2, 1, 2)
plt.setp(plt.gca().get_xticklabels(), horizontalalignment='center')
plt.xticks(range(len(aList)), aList, FontProperties=font_set, rotation=60)
plt.bar(pList, pValue)
plt.ylim(0, 1300)
plt.savefig("./pic/Weather.png", bbox_inches='tight')
plt.show()

print('Saved')
```

4.3.3 Main.py

```
import codecs
import requests, os
from bs4 import BeautifulSoup
from lxml import etree
import re
from openpyxl import Workbook, load_workbook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator, FuncFormatter
from pandas import DataFrame
```

```
import math
import time

import weatherCrawler
import DataAnalysis

mainurl = 'http://www.tianqihoubao.com'
url = 'http://www.tianqihoubao.com/lishi'
encoding = 'gbk'
head = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)'
                      ' AppleWebKit/605.1.15 (KHTML, like Gecko)'
                      ' Version/12.0.1 Safari/605.1.15',
        'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',}
filename = 'WeatherHistory'
req, reqState = weatherCrawler.urlAnalysis(url, filename)
maxCityNum = 1

if reqState == 200:
    weatherCrawler.main(mainurl, reqState, req, maxCityNum, filename, encoding)
else:
    print(reqState)

xlsxFile = pd.ExcelFile('./WeatherHistory/北京.xlsx')#分析不同的 excel 需要不同
的文件名
DataAnalysis.tempPlot(xlsxFile)
```

5. 代码测试

完成日期：2019 年 05 月 09 日
代码运行前：

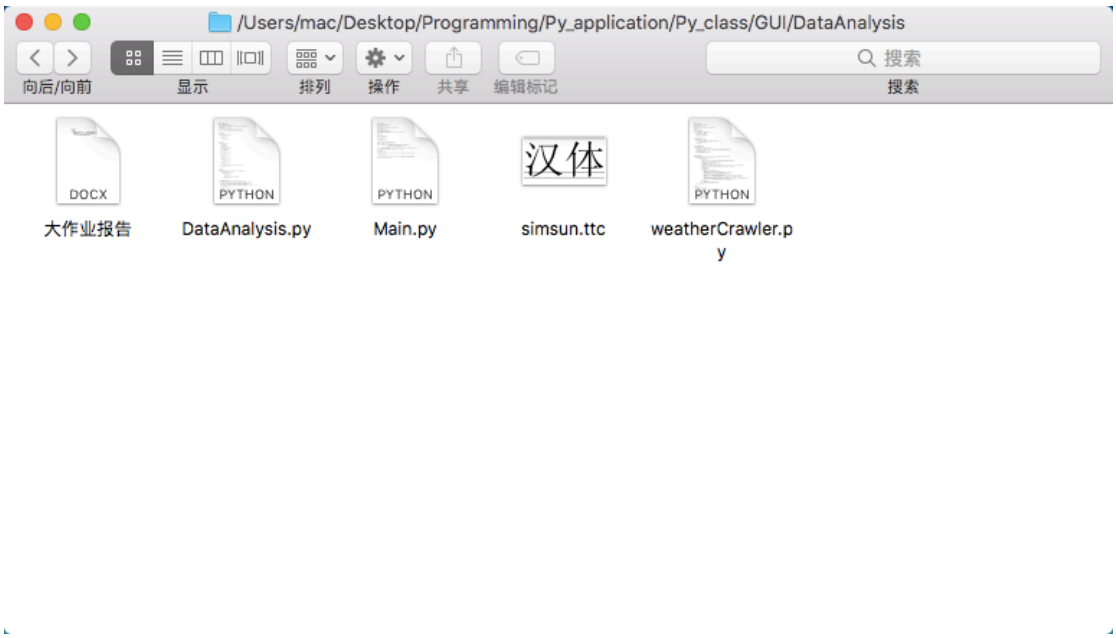

图 3 运行前文件截图

图 4 运行当天网页截图 1

图 5 运行当天网页截图 2

图 6 运行当天网页截图 3

```
/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/mac/Desktop/Programming/Py_application/Py_class/GUI/DataAnalysis/Main.py
Url Request Accepted
http://www.tianqihoubao.com/lishi/beijing.html
北京
['/lishi/beijing/month/201101.html', '/lishi/beijing/month/201102.html', '/lishi/beijing/month/201103.html', '/lishi/beijing/month/201104.html', '/lishi/beijing/month/201105.html', '/li
['2011年1月北京天气', '2011年2月北京天气', '2011年3月北京天气', '2011年4月北京天气', '2011年5月北京天气', '2011年6月北京天气', '2011年7月北京天气', '2011年8月北京天气', '2011年9月北京天气', '2011年10月
2011年1月北京天气 already exist
2019-05-08 23:14:42
2011年2月北京天气 already exist
2019-05-08 23:14:47
2011年3月北京天气 already exist
2019-05-08 23:14:53
2011年4月北京天气 already exist
2019-05-08 23:15:12
2011年5月北京天气 already exist
2019-05-08 23:15:16
2011年6月北京天气 already exist
2019-05-08 23:15:18
2011年7月北京天气 already exist
2019-05-08 23:15:22
2011年8月北京天气 already exist
2019-05-08 23:15:30
2011年9月北京天气 already exist
2019-05-08 23:15:37
2011年10月北京天气 already exist
2019-05-08 23:15:41
2011年11月北京天气 already exist
2019-05-08 23:15:46
2011年12月北京天气 already exist
2019-05-08 23:15:55
2012年1月北京天气 already exist
2019-05-08 23:16:00
2012年2月北京天气 already exist
2019-05-08 23:16:04
2012年3月北京天气 already exist
2019-05-08 23:16:11
2012年4月北京天气 already exist
2019-05-08 23:16:15
2012年5月北京天气 already exist
2019-05-08 23:16:18
2012年6月北京天气 already exist
2019-05-08 23:16:21
2012年7月北京天气 already exist
2019-05-08 23:16:23
2012年8月北京天气 already exist
2019-05-08 23:16:29
2012年9月北京天气 already exist
2019-05-08 23:16:33
2012年10月北京天气 already exist
2019-05-08 23:16:38
2012年11月北京天气 already exist
2019-05-08 23:16:45
2012年12月北京天气 already exist
```

图 7 运行结果图 1

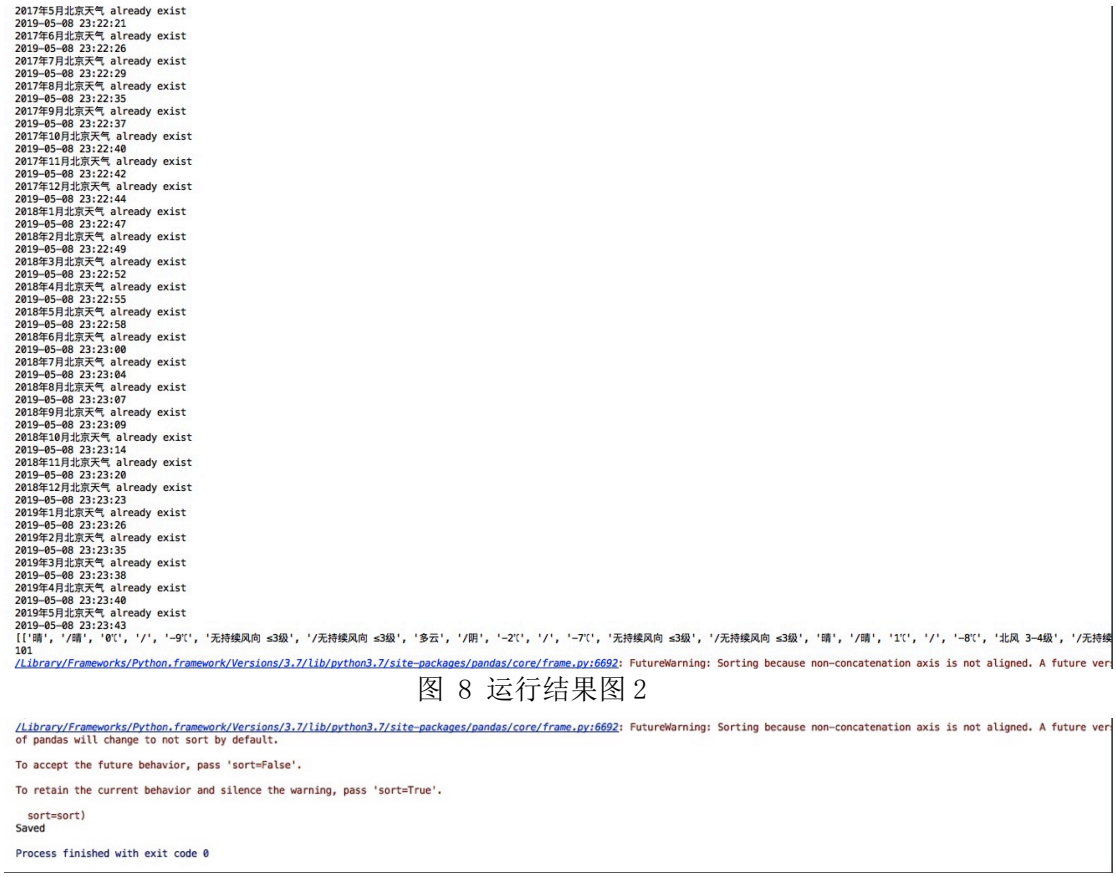

图 8 运行结果图 2

图 9 运行结果图 3

注：文件已存在是为了节省时间，先跑了一次 WeatherCrawler 的结果

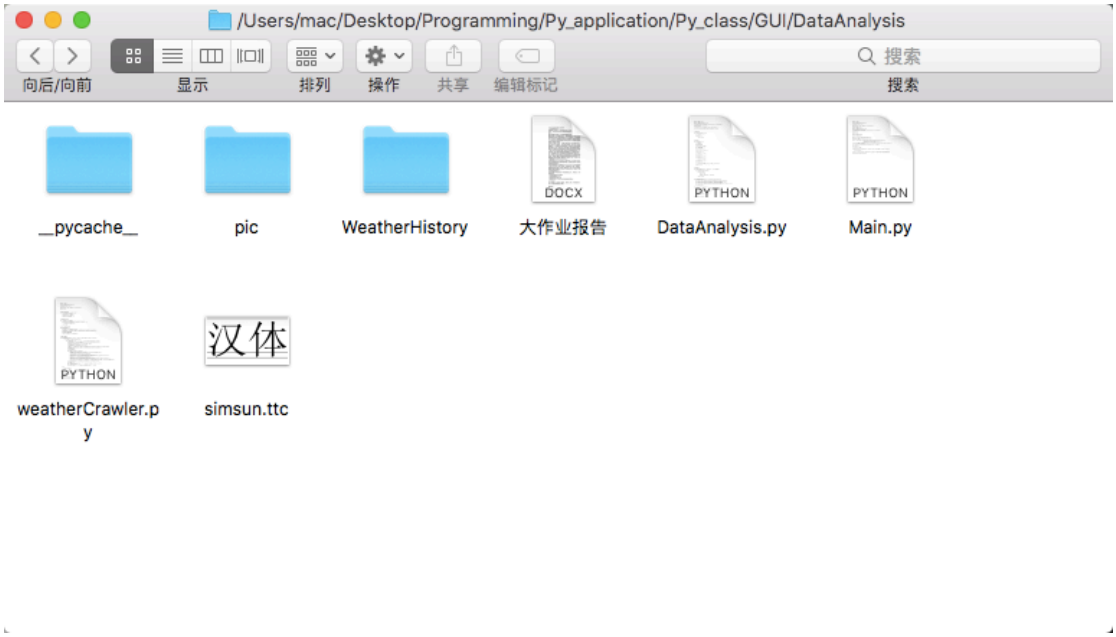

图 10 爬取后文件结果

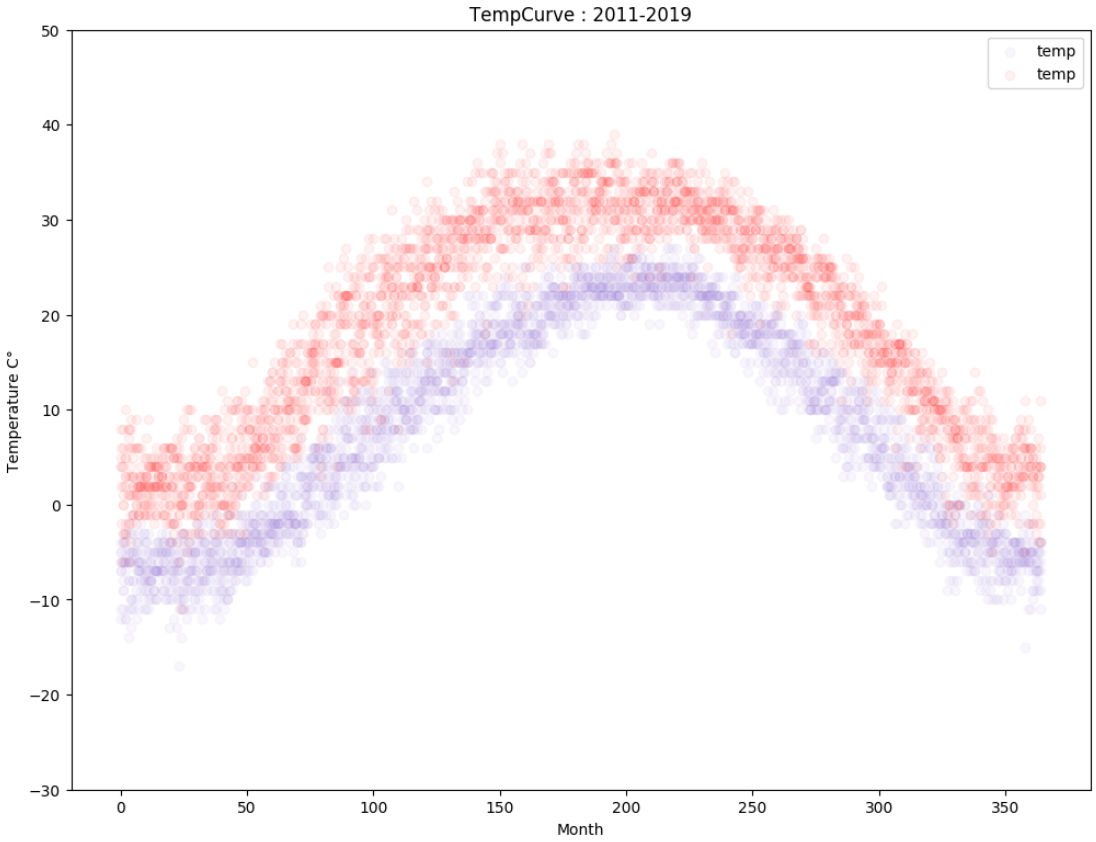

图 11 最高/最低气温散点图

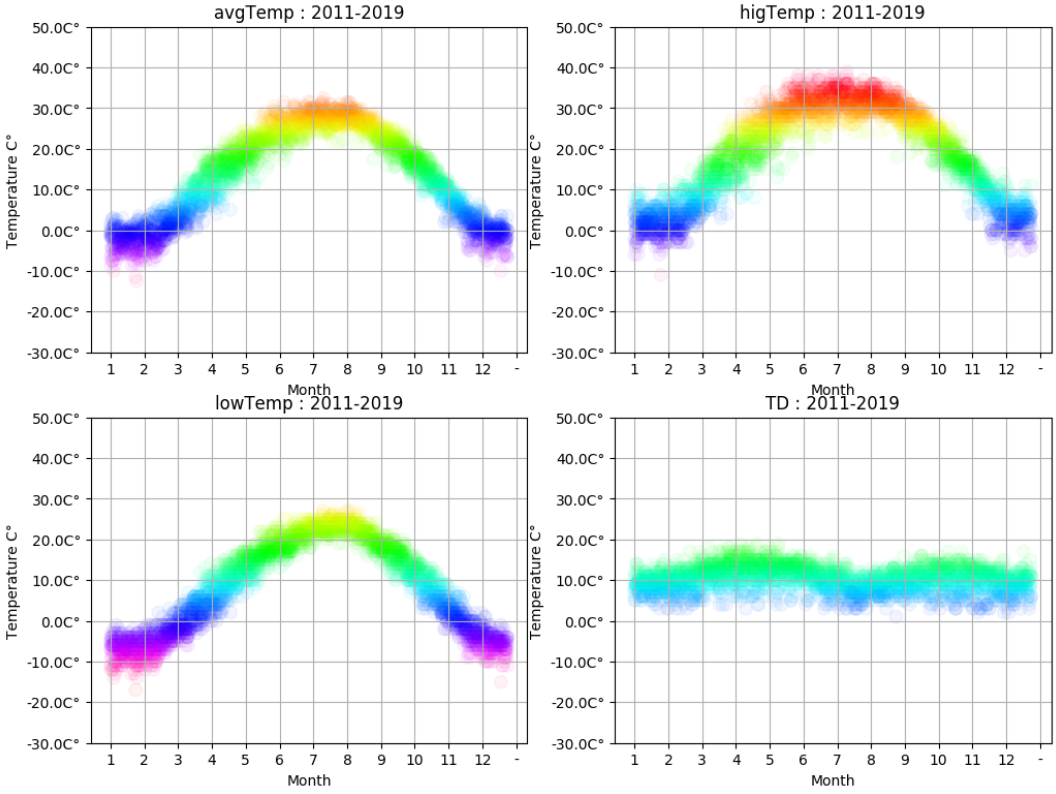

图 12 平均气温、最高温、最低温、温度差色度散点图

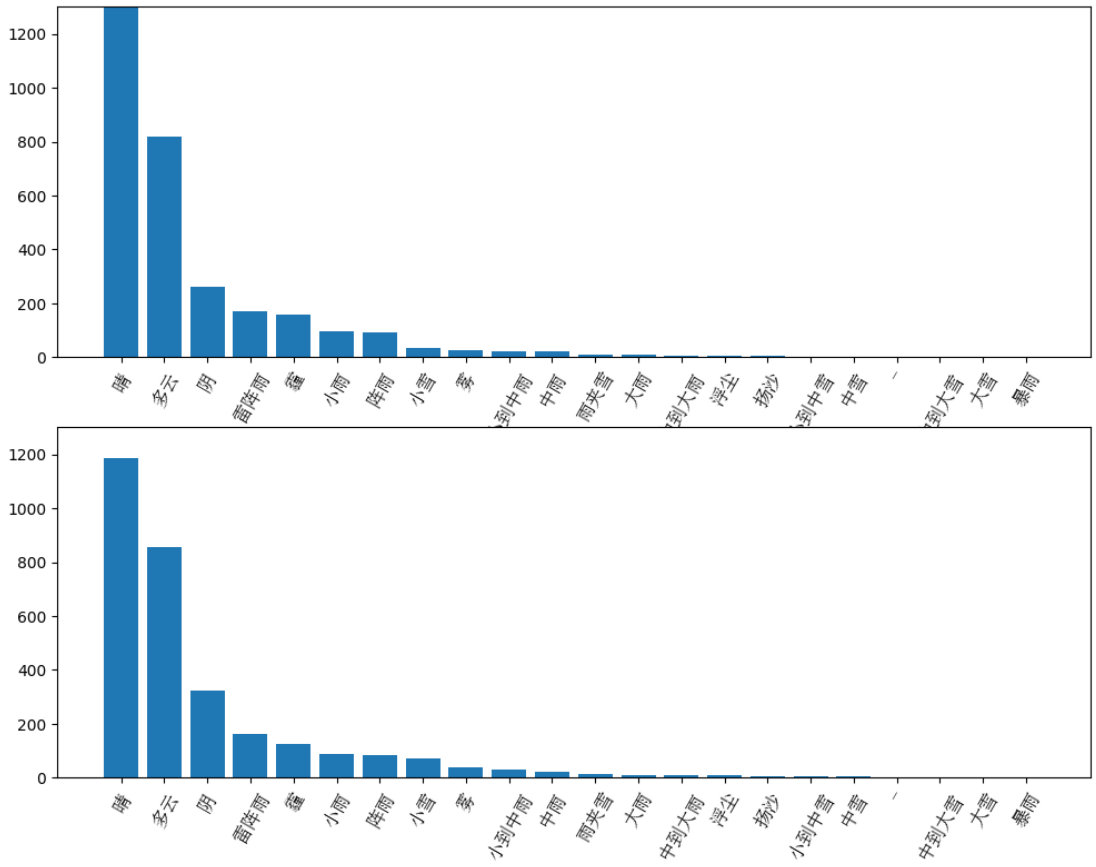

图 13 各类气象 am/pm 出现情况图

6. 结论

北京地区常年每日温差平均值为 10℃左右，且春秋温差大，冬夏温差小，所以春秋应该注意增减衣物，以应对昼夜温差变化。

北京地区历年每日平均气温浮动有 10℃左右，可以看出北京地区气候多变，夏季也有凉爽的时候，冬季也有暖和的时候。

北京地区冬天气温低，夏天气温高，夏天最高气温浮动可达 20℃，而冬季最低气温浮动仅有 10℃，可见北京地区夏天十分炎热，而冬季寒冷程度比较稳定。

未来的方向：本系统只要修改一两个参数即可爬取分析另一个城市，可以增加一个 GUI，输入所需城市即可得到相应结论；

不足：没有时间添加多线程，爬取速度较为缓慢。

7. 致谢

首先，要感谢我的老师的悉心教导，对论文的主题结构、写作细节和论文格式都给予了细致的点评，在此表示衷心的感谢。

感谢我的家人，感谢他们一直以来对我的照顾，让我在轻松快乐的环境中全身心的投入到研究中来。

由于本人学识有限，文中肯定有偏颇和不妥之处，恳切希望老师同学们予以批评和指正。

8. 参考文献

[1]司林青. 基于历史数据的天气预报系统设计与实现[D].黑龙江大学,2018.