爬虫，主讲：汤小洋

一、爬虫简介

1. 爬虫是什么？

爬虫，称为网页蜘蛛或网络机器人，用于自动获（爬）取互联网上的信息，本质上就是一段代码

任何一门高级开发语言都可以实现爬虫，并不只有Python

2. 实现原理

通过代码，模拟浏览器向服务器发送HTTP或HTTPS请求，然后对服务器响应的结果进行处理，从中获取想要的数据

三步走：

- 获取数据：发送请求并接收响应结果
- 处理数据：对响应结果进行处理，筛选出有效数据
- 存储数据：将有效数据存储起来

二、基本用法

1. 获取数据

使用urllib模块模拟浏览器发送请求

```python
# 获取数据
def get_data():
    url = 
'https://search.51job.com/list/070200,000000,0000,00,9,99,java%25E5%25BC%2580%25E5%258F
%2591,2,1.html'
    # 创建Request对象，指定url和请求头
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/56.0.2924.87 Safari/537.36'
    }
    req = request.Request(url, headers=headers)
    response = request.urlopen(req)
    # print(type(response))  # HTTPResponse类型
    # print(response.getcode())  # 响应状态码
    # print(response.info())

    if response.getcode() == 200:
        data = response.read()  # 读取响应结果
        # print(type(data)) # bytes类型
        data = str(data, encoding='gbk')  # 转换为str
        # print(data)

        # 将数据写入文件中
        with open('index.html', mode='w', encoding='gbk') as f:
            f.write(data)
```

2. 处理数据

三种方式：

- 字符串解析

  使用字符串+正则表达式

- 使用XPath

  XPath是一门在XML文档中查找信息的语言，用来在XML文档中对元素和属性进行遍历。

  使用Chrome浏览器的开发人员工具，获取XPath

- 使用第三方模块BeautifulSoup

  Beautiful Soup 是一个可以从HTML或XML文件中提取数据的Python库

  安装 `pip install beautifulsoup4`

```
# 处理数据
def parse_data():
    with open('index.html', mode='r', encoding='gbk') as f:
        html = f.read()

    #  创建BeautifulSoup实例，解析html数据
    bs = BeautifulSoup(html, 'html.parser')  # 指定使用html解析器parser

    '''
    查找数据
    '''
    # 1.find()方法，获取第一个匹配的标签
    # div = bs.find('div')
    # print(div)
    # print(type(div))  # Tag类型

    # 2.find_all()方法，获取所有匹配的标签
    # metas = bs.find_all('meta')  # 返回的是集合
    # print(metas[0])
    # print(bs.find_all(id='hello'))  # 根据id获取，返回的是集合
    # print(bs.find_all(class_='itany'))  # 根据class获取

    # 3.select()方法，使用CSS选择器来获取元素
    # print(bs.select('#hello'))
    # print(bs.select('.itany'))
    # print(bs.select('p#world span'))
    # print(bs.select('[title]'))

    # 4.get_text()方法，获取Tag中的文本
    # value = bs.select('#hello')[0].get_text(strip=True)
    # print(len(value))
    # print(value)

    # 获取职位信息
    divs = bs.select('#resultList .el')
    result = []
    for div in divs[1:]:
        title = div.select('.t1')[0].get_text(strip=True)
        company = div.select('.t2')[0].get_text(strip=True)
        addr = div.select('.t3')[0].get_text(strip=True)
        salary = div.select('.t4')[0].get_text(strip=True)
        pubDate = div.select('.t5')[0].get_text(strip=True)
        # print(title, company, addr, salary, pubDate)
        row = {
            'title': title,
            'company': company,
            'addr': addr,
            'salary': salary,
            'salary': salary,
            'pubDate': pubDate
        }
        result.append(row)
    return result
```

3. 存储数据

3.1 存储MySQL

```
# 存储数据到MySQL
def save_to_mysql(data):
    config = {
        'host': 'localhost',
        'port': 3306,
        'user': 'root',
        'password': '',
        'database': 'python',
        'charset': 'utf8'
    }
    conn = pymysql.connect(**config)
    cursor = conn.cursor()
    sql = '''
        insert into t_job
          (title, company, addr, salary, pubDate)
        values
          (%(title)s,%(company)s,%(addr)s,%(salary)s,%(pubDate)s)
    '''
    cursor.executemany(sql, data)
    conn.commit()

    cursor.close()
    conn.close()
```

3.2 存储到Excel

使用openpyxl模块操作Excel

安装openpyxl：`pip install openpyxl`

工作薄Workbook

工作表Sheet

单元格Cell

```python
# 存储数据到Excel
def save_to_excel(data):
    # 创建工作薄Workbook
    book = Workbook()

    # 创建工作表Sheet
    sheet = book.create_sheet('南京Java招聘信息', 0)

    # 向工作表中添加数据
    sheet.append(['职位名', '公司名', '工作地点', '薪资', '发布时间'])
    for item in data:
        row = [item['title'], item['company'], item['addr'], item['salary'],
item['pubDate']]
        sheet.append(row)

    # 输出保存
    book.save('51job.xlsx')
```

3.3 存储到Redis

安装redis库： `pip install redis`

```python
# 存储数据到Redis
def save_to_redis(data):
    config = {
        'host': '192.168.2.30',
        'port': 6379,
        'charset': 'utf8'
    }
    r = redis.Redis(**config)
    # r.set('name', 'tom')
    for item in data:
        r.lpush('jobs', item)


# 从Redis中读取数据
def read_from_redis():
    config = {
        'host': '192.168.2.30',
        'port': 6379,
        'charset': 'utf8',
        'decode_responses': True  # 读取时解码
    }
    r = redis.Redis(**config)
    print(r.lrange('jobs', 0, -1))
```

三、处理JSON数据

```
from urllib import request
import json


def get_data():
    url = 'https://movie.douban.com/j/search_subjects?
type=movie&tag=%E7%83%AD%E9%97%A8&sort=recommend&page_limit=400&page_start=0'
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/56.0.2924.87 Safari/537.36'
    }
    req = request.Request(url, headers=headers)
    response = request.urlopen(req)
    if response.getcode() == 200:
        result = response.read()
        # print(type(result))  # bytes类型
        return result


def parse_data(html):
    # 将字符串形式的json转换为dict字典
    data = json.loads(html)
    # print(type(data), data)
    movies = data['subjects']
    for movie in movies:
        print(movie['title'], movie['rate'])


if __name__ == '__main__':
    parse_data(get_data())
```

四、爬虫应用

步骤：

1. 获取数据
2. 处理数据
3. 存储数据
4. 数据可视化

1. 电影评论数据分析

```
from urllib import request
import json
from datetime import datetime, timedelta
import time


# 获取数据
def get_data(url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 9_1 like Mac OS X)
AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13B143 Safari/601.1'
    }
    req = request.Request(url, headers=headers)
    response = request.urlopen(req)
    if response.getcode() == 200:
        return response.read()


# 处理数据
def parse_data(html):
    data = json.loads(html)['cmts']
    comments = []
    for item in data:
        comment = {
            'id': item['id'],
            'nickName': item['nickName'],
            'cityName': item['cityName'] if 'cityName' in item else '',  # 处理cityName
不存在情况
            'content': item['content'].replace('\n', ' '),  # 处理评论内容换行的情况
            'score': item['score'],
            'startTime': item['startTime']
        }
        comments.append(comment)
    return comments


# 存储数据到文本文件
def save_to_txt():
    start_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  # 当前时间
    end_time = '2018-08-10 00:00:00'  # 结束时间
    while start_time > end_time:
        url = 'http://m.maoyan.com/mmdb/comments/movie/1203084.json?
_v_=yes&offset=0&startTime=' + start_time.replace(
            ' ', '%20')
        try:
            html = get_data(url)
        except:
            time.sleep(1)
            html = get_data(url)
        else:
            time.sleep(0.1)

        comments = parse_data(html)
        print(comments)

        start_time = comments[14]['startTime']  # 末尾评论时间
        start_time = datetime.strptime(start_time, '%Y-%m-%d %H:%M:%S') -
timedelta(seconds=1)  # 向前减 1 秒，防止获取到重复数据
        start_time = datetime.strftime(start_time, '%Y-%m-%d %H:%M:%S')

        for item in comments:
            with open('comments.txt', mode='a', encoding='utf-8') as f:
                f.write(str(item['id']) + ',' + item['nickName'] + ',' +
item['cityName'] + ',' + item[
                    'content'] + ',' + str(item['score']) + ',' + item['startTime'] +
'\n')


if __name__ == '__main__':
    # url = 'http://m.maoyan.com/mmdb/comments/movie/1203084.json?
_v_=yes&offset=15&startTime=2018-09-01%2011%3A10%3A00'
    # comments = parse_data(get_data(url))
    # print(comments)
    save_to_txt()
```

2. 数据可视化

pyecharts类库

2.1 粉丝位置分布

```python
from collections import Counter
from pyecharts import Geo
import json
from pyecharts import Bar


def render():
    # 获取所有城市信息
    cities = []
    with open('comments.txt', mode='r', encoding='utf-8') as f:
        rows = f.readlines()
        for row in rows:
            city = row.split(',')[2]
            if city != '':
                cities.append(city)

    # 对城市数据和坐标文件中的地名进行处理
    handle(cities)

    # 统计每个城市出现的次数
    # data = []  # [('南京',25),('北京',59)]
    # for city in set(cities):
    #     data.append((city, cities.count(city)))
    data = Counter(cities).most_common()

    # 根据城市数据生成地理坐标图
    geo = Geo(
        "《一出好戏》粉丝位置分布",
        "数据来源：猫眼",
        title_color="#fff",
        title_pos="center",
        width=1200,
        height=600,
        background_color="#404a59",
    )
    attr, value = geo.cast(data)
    geo.add(
        "",
        attr,
        value,
        visual_range=[0, 3500],
        visual_text_color="#fff",
        symbol_size=15,
        is_visualmap=True,
    )
    geo.render('粉丝位置分布.html')


    # 根据城市数据生成柱状图
    # 根据城市数据生成柱状图
    cities_top20 = Counter(cities).most_common(20)  # 返回出现次数最多的20条
    bar = Bar("《一出好戏》粉丝来源排行榜TOP20", '数据来源：猫眼', title_pos='center',
width=1200, height=600)
    attr, value = bar.cast(cities_top20)
    bar.add("", attr, value)
    bar.render('粉丝来源排行榜-柱状图.html')


# 处理地名数据，解析坐标文件中找不到地名的问题
def handle(cities):
    with open(
            'C:/Users/User/PycharmProjects/python-spider/venv/Lib/site-
packages/pyecharts/datasets/city_coordinates.json',
            mode='r', encoding='utf-8') as f:
        data = json.loads(f.read())  # 将str转换为dict

    # 循环判断处理
    data_new = data.copy()  # 复制一份地名数据
    for city in set(cities):
        count = 0
        for k in data:
            count += 1
            if k == city:
                break
            if k.startswith(city):  # 处理简写的地名，如南京市 简写为 南京
                data_new[city] = data[k]
                break
            if k.startswith(city[0:-1]) and len(city) >= 3:  # 处理行政变更的地名，如溧水县
改为 溧水区
                data_new[city] = data[k]
                break
        # 处理不存在的情况
        if count == len(data):
            while city in cities:
                cities.remove(city)
    # print(len(data), len(data_new))

    # 写入覆盖坐标文件
    with open(
            'C:/Users/User/PycharmProjects/python-spider/venv/Lib/site-
packages/pyecharts/datasets/city_coordinates.json',
            mode='w', encoding='utf-8') as f:
        f.write(json.dumps(data_new, ensure_ascii=False))  # 将dict转换为str，指定
ensure_ascii=False支持中文


if __name__ == '__main__':
    render()
    render()
```

2.2 评价星级

```
from pyecharts import Pie

#  获取评论中所有评分
rates = []
with open('comments.txt', mode='r', encoding='utf-8') as f:
    rows = f.readlines()
    for row in rows:
        rates.append(row.split(',')[4])

# print(rates)

# 定义星级
attr = ['五星', '四星', '三星', '二星', '一星']
value = [
    rates.count('5') + rates.count('4.5'),
    rates.count('4') + rates.count('3.5'),
    rates.count('3') + rates.count('2.5'),
    rates.count('2') + rates.count('1.5'),
    rates.count('1') + rates.count('0.5')
]
# print(value)

pie = Pie("《一出好戏》评分星级", title_pos='center', width=900)
pie.add("", attr, value, is_label_show=True, is_legend_show=False)
pie.render('电影评分-饼图.html')
```

2.3 词云图

jieba(结巴)是一个强大的分词库,完美支持中文分词

Matplotlib 是一个Python的 2D绘图库，可以生成绘图，直方图，功率谱，条形图，错误图，散点图等

wordcloud基于Python的词云生成类库,很好用,而且功能强大

```python
import jieba
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS

# 获取所有评论内容
comments = []
with open('comments.txt', mode='r', encoding='utf-8') as f:
    rows = f.readlines()
    for row in rows:
        comment = row.split(',')[3]
        if comment != '':
            comments.append(comment)

# 设置分词
comment_after_split = jieba.cut(str(comments), cut_all=False)
words = ' '.join(comment_after_split)  # 以空格进行拼接
# print(words)

# 设置屏蔽词汇
stopwords = STOPWORDS.copy()
stopwords.add('电影')
stopwords.add('一出')
stopwords.add('好戏')
stopwords.add('有点')

# 导入背景图
bg_image = plt.imread('love.jpg')

# 设置词云参数
wc = WordCloud(width=1024, height=768, background_color='white', mask=bg_image,
stopwords=stopwords, max_font_size=400,
               random_state=50,font_path='STKAITI.TTF')
# 将分词后数据导入云图
wc.generate_from_text(words)
#  绘制图像
plt.imshow(wc)
plt.axis('off')  # 不显示坐标轴
plt.show()  # 显示图像
# 保存图像到文件
wc.to_file('词云图.jpg')
```