摘要

本项目采用爬虫、前端开发（HTML5+CSS+JavaScript）、后端开发（Python）、框架处理（Flask）、API 接入（高德地图 API）、数据库编程（SQLite3）、多线程处理等技术，实现了从目标网页上获取对应数据并极大地对数据进行了可视化处理，极大地优化了用户客户端的体验。再者，我们使用爬虫可不仅仅满足于呈现爬虫的数据结果，而且还要将爬取的数据结果以丰富多彩的形式展现出来，达到生动形象的要求。其主要功能包括豆瓣网 TOP250 电影排名榜单显示、豆瓣网查询单一城市上映电影院线显示、全国院线数量柱状图可视化显示于地图、TOP250 电影数据转存数据库并分析已爬取的数据以统计评分、生成概述爬取数据的词云并允许用户根据自己的图片样式喜好保存词云图片。项目特点是界面美观大方、操作简单明了、运行流畅快速、数据更新准时、功能实用新颖。

关键字：爬虫；Flask；API；HTML5；SQLite3。

目录

1. 绪论 1
   1.1. 需求分析 1
   1.2. 开发环境 1
2. 设计与实现 1
   2.1. 概要设计 1
      2.1.1. 明确需求 1
      2.1.2. 建立模型 1
   2.2. 详细设计与实现 2
      2.2.1. 后端开发 2
      2.2.2. 前端开发 5
      2.2.3. 数据库开发 7
      2.2.4. Flask、Ajax&jQuery 引入 7
      2.2.5. API 接入 8
      2.2.6. 词云 9
3. 系统测试 9
   3.1.测试日志 9
   3.2.演示系统截图 10
4. 总结 15
   4.1.系统特点 15
   4.2.开发过程中遇到的问题 15
   4.3.收获总结 15
5. 参考文献 16

1.绪论

1.1. 需求分析

随着互联网的迅速发展，万维网成为大量信息的载体，如何有效地提取并利用这些信息成为一个巨大的挑战。为了短时间内能够准确的获得大量指定的信息，网络爬虫应运而生。本次项目以一名电影爱好者的角度进行编写，对于热门网络影院（如豆瓣）的指定内容进行爬取，并且对内容信息进行处理，收集，最后完成数据可视化展示。

1.2. 开发环境

Python3：PyCharm
HTML5：WebStorm
CSS3：WebStorm
JavaScript：WebStorm
SQLite3：PyCharm
操作系统：Windows10/Windows11

2.设计与实现

2.1.概要设计

2.1.1.明确需求：

爬虫获取豆瓣电影数据并进行最大程度的可视化分析。

2.1.2 建立模型：

模型为 Web 端，网页概述框架如图 2.1：

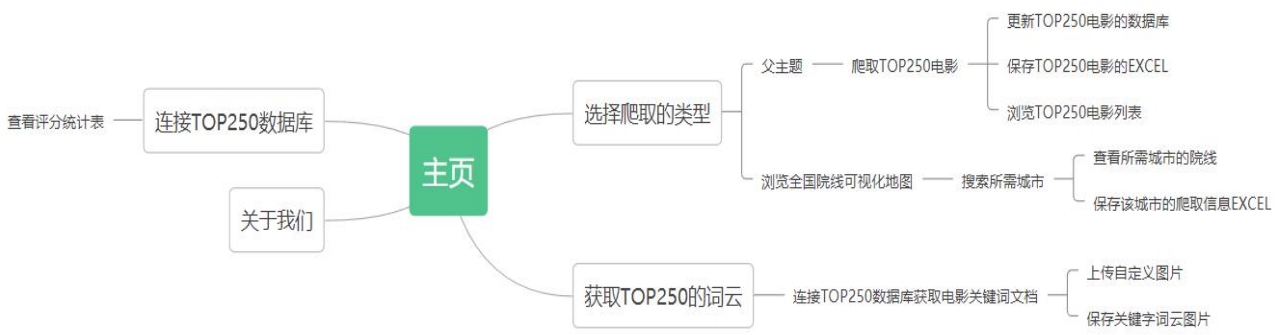

图 2.1 网页概述框架

2.2 详细设计与实现

2.2.1.后端开发

后端框架如图 2.2.1 所示：

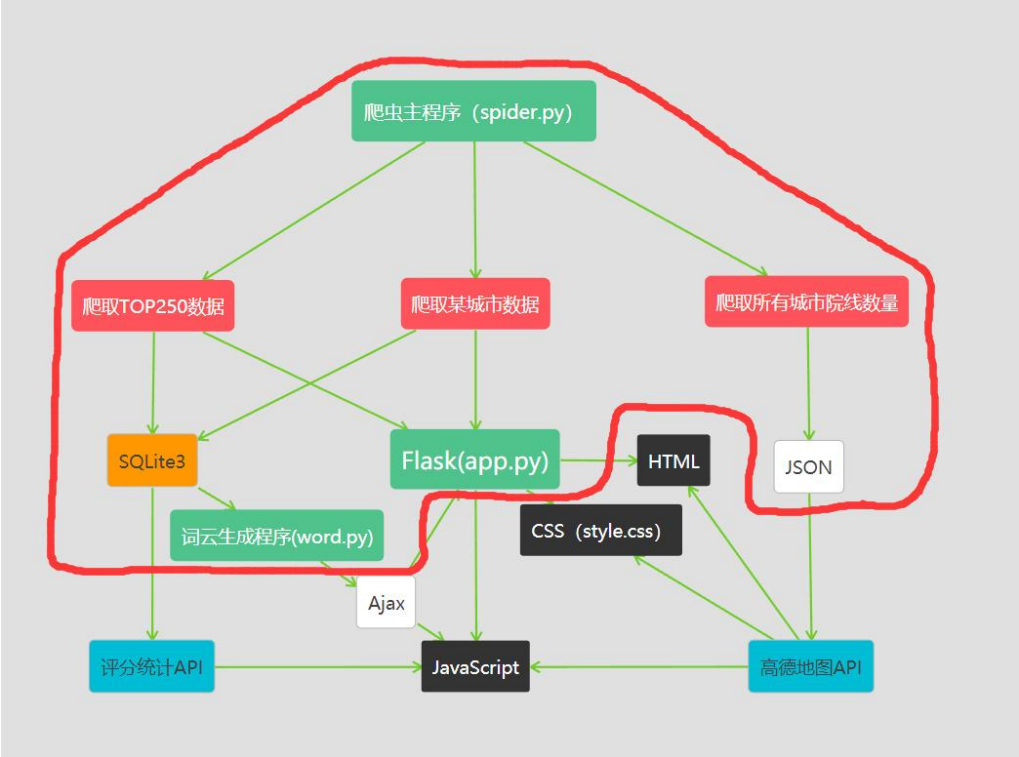

图 2.2.1 后端框架示意图

#关于 BeautifulSoup4

BeautifulSoup 是一个可以从 HTML 或 XML 文件中提取数据的 Python 库，它能够通过你喜欢的转换器实现惯用的文档导航、查找、修改文档的方式。通俗点说，就是能帮你从一个文档中提取出符合某种特征的内容；一种情况下，这个文档可以是我们给他的，这种情况多半是我们训练自己水平的时候使用的。另一种就是这种文档是你从网络上使用爬虫爬来的，然后从里面提取所有的图片，链接，标签，评分，简介，评价人数等等，很有价值和成就感。

（1）数据获取

先对指定网页进行爬取，获得未加工的初始数据。同时为了防止 python 爬虫被网站认定为是爬虫程序，我们需要通过更改 head 中的“User-Agent”来实现对其的伪装操作，让网站认为我们是用浏览器进行访问的。

askURL1 用于获得豆瓣 TOP250 电影的页面数据

```
def askURL1(url)
```

askURL2 用于获得某城市上映电影的页面数据

```
def askURL2(url)
```

（2）数据解析

将之前存入 val 的未加工数据引入，并一一对其进行数据解析，并且在解析完成后凭借正则表达式获取指定内容。

multi_thread 用于多线程操作，快速爬取数据

```
def multi_thread(usl1):
    return threads
```

getData1 用于处理豆瓣 TOP250 数据

```
def getData1(baseurl):
    return datalist
```

getData2 用于处理某城市上映电影的页面数据

```
def getData2(baseurl):
    return datalist
```

（3）数据储存

对爬取的指定内容数据进行储存，方便之后进行可视化数据的展示，数据存储不仅仅可以以 excel 表格的形式，而且可以导入 sqlite 库，进行数据库编程将爬取到的电影信息存入数据库以供后续其他操作。其中 excel 表的存储方便了用户对于全部电影信息的了解，而数据库对于信息的存储则方便了后续评分数据的导出等一系列可视化操作。

①init_tb1 函数创建一个表用来保存从网站爬取的 top250 的电影

```
def init_tb1(dbpath)
```

②init_tb2 函数创建一个表用来保存从网站爬取的当前正在上映的电影

```
def init_tb2(dbpath)
```

③saveData1 用于储存豆瓣 TOP250 的电影数据于 excel 表

```
def saveData1(datalist,savepath):
book.save(savepath)
```

④saveData2 用于存储某城市上映电影的数据于数据库

```
def saveData2(datalist,savepath):
book.save(savepath)
```

⑤saveData1DB1 用于储存豆瓣 TOP250 电影的数据于数据库

```
def saveData1DB1(datalist,dbpath)
```

⑥saveData2DB2 用于储存某城市上映电影的数据于数据库

```
def saveData2DB2(datalist,dbpath)
```

⑦高德地图 API 数据修改

通过 JSON 文件导入高德地图组件的数据是由后台进行更新的，并不是随着爬取数据的更新而更新，由后台人员定时进行操作，操作函数如下：

```
def updateCitiesData()
```

2.2.2. 前端开发

前端框架如图 2.2.2 所示：

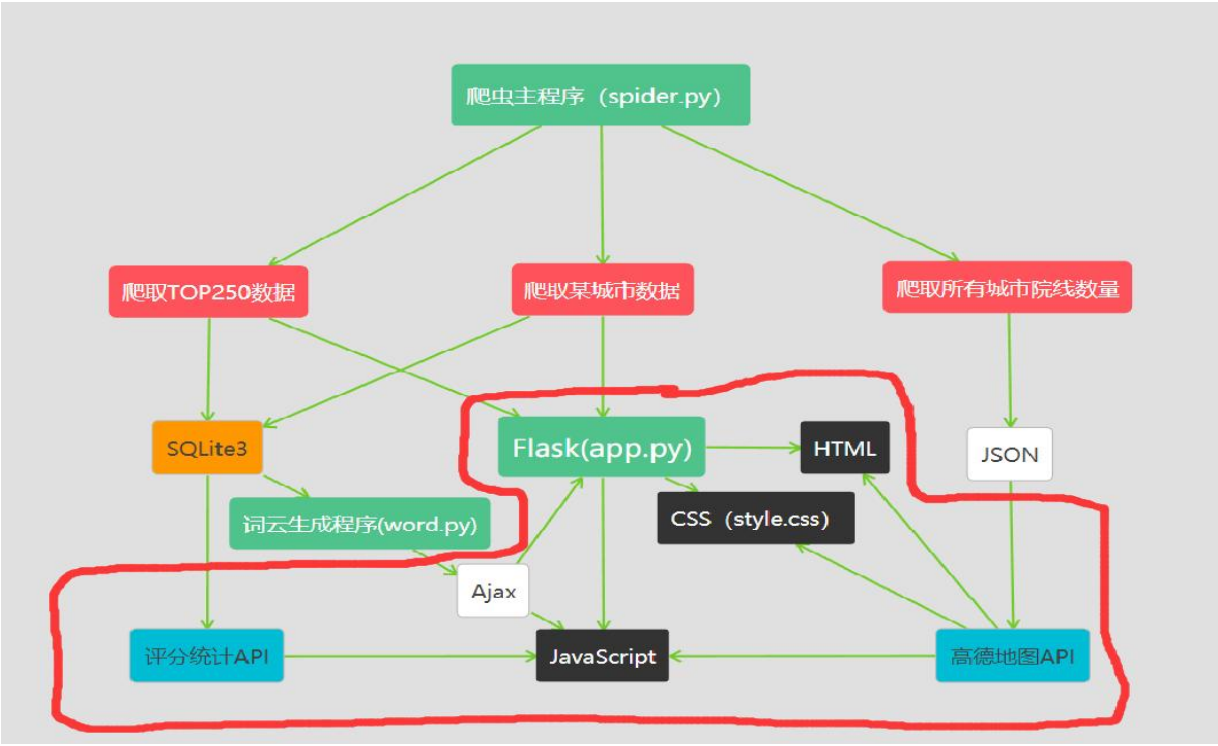

图 2.2.2 前端框架示意图

（1）要设计一个简洁明了、操作简单的界面，就要将界面内容尽量精简，突出主题操作对象。于是可以这样设计 HTML 网页框架：网站框架包含顶部导航栏、底部信息栏、浮动置顶栏。

```
<title>豆瓣影视数据分析</title>
    <body>
            <header id="header">
                    <div class="container">
                    </div>
            </header><!-- End Header -->
            <footer id="footer">
            </footer><!-- End Footer -->

    </body>
```

（2）丰富却简洁的用户交互有利于提升网页的整体品质，为此我们在 Python 开发端引入了轻量型网页开发框架 Flask，以便快速而又智慧地生成美观大方地网页，详见（5.Flask、Ajax 框架引入）。

（3）前端对数据的处理也是重中之重，要把前端表单 INPUT 标签的值送入后端可不是一件容易事，为此我们采用了 URL 传递参数的方式，例如./app.py 中，使用了 request.url 将前端的输入数据送入 Flask 框架中处理。

本项目实现了输入城市名称以爬取该城市数据的功能，在 cities.html 中：

```
<input class = "search" type="text" name="cityname" placeholder="请输入城市名称（支持汉字和完整拼音）">
<script>
function open_details(){
    window.open("/citylist?cityname=" + document.forms["subform"]["cityname"].value);
}

</script>
```

（4）在 app.py 中，承载着 Flask 框架，处理弹出的窗口"/citylist?cityname=" + document.forms["subform"]["cityname"].value 的内容。而如果使用 cityname =

request.url 就可以获取表单中输入城市的信息。对爬取数据进行表格化处理，这里我们使用 Flask 模块的"{{}}"访问 app.py 的 movie 变量（movie 是 app.py 中存放爬取数据的列表）

（5）前端还要进行图片文件上传保存到项目根目录的处理，为此引入了 Ajax&jQuery 模块进行处理，详见（2.2.4Flask、Ajax&jQuery 引入）

（6）CSS3 的导入，详见项目./static/assets/css/style.sss，这里不一一列举对页面进行的风格编辑处理。

（7）JavaScript 脚本的灵活运用，使网页的内容呈现形式更加多样。为此我们引入了评分统计功能和高德地图显示的 API，引入代码如下：

cities.html：（引入高德地图 API 的使用 div）

```
<div id="map">
```

score.html：（引入评分统计功能 API 的使用 div）

```
<div id="main" style="width: 100%;height:500px;"></div>
    <script type="text/javascript">
    </script>
</div>
```

2.2.3.数据库开发

（1）saveData1DB1 用于储存豆瓣 TOP250 电影的数据于数据库，

```
def saveData1DB1(datalist,dbpath)
```

（2）saveData2DB2 用于储存某城市上映电影的数据于数据库

```
def saveData2DB2(datalist,dbpath)
```

（3）数据库存储爬取的电影信息：

```
def saveData2(datalist,savepath)
```

（4）连接数据库 movie250 并查询高频词汇，以生成词云。

```
def makeWordcloud(image_name)
```

（5）连接数据库 movie250 并查询电影评分信息，用于评分表展示。

```
@app.route('/score')
def score():
return render_template('score.html', movieScore=movieScore, num=num)
```

2.2.4.Flask、Ajax&jQuery 引入

（1）Flask 框架概述：

Flask 是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug ，模板引擎则使用 Jinja2 。Flask 使用 BSD 授权。Flask 也被称为“microframework” ，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。我们使用此框架来实现数据的可视化。

我们根据 Flask 的轻量级特性，借此来将编写好的前端和数据进行 web 的实现。按照惯例， 模板和静态文件存放在应用的 Python 源代码树的子目录中，名称分别为 templates 和 static。在静态子目录下建立 assets 文件夹，其中放入 JS 和 CSS 以及 JSON 和 img（图片来源文件夹）等前端相关的代码，JS 负责前端交互，CSS 负责样式和美化。

templates 目录下存放已经编写好的 HTML 代码，用来描述我们的网页，再通过 app.py 即可实现我们的个性化 web。

app.py 的作用是一个 Flask 框架实现的实例程序，当客户端点击时，发出 HTTP 请求，Web 服务器使用 WSGI 协议把来自客户端的请求发送到 Flask 程序实例中。Flask 使用 Werkzeug 来做路由分发，根据每个 URL 请求，找到对应的视图函数，路由通过装饰器实现，获取到数据后，把数据传入到 HTML 模板中，然后渲染 HTTP 响应数据，由 Flask 将响应数据返回给浏览器，即可实现我们的不同网页显示结果，也就是我们数据可视化的结果。

实例如下：

```
@app.route('/first')
def first():
  return index()
```

（2）Ajax&JQuery 概述：

Ajax 即 Asynchronous Javascript And XML（异步 JavaScript 和 XML），用来描述一种使用现有技术集合的“新”方法，包括: HTML 或 XHTML, CSS, JavaScript, DOM, XML, XSLT, 以及最重要的 XMLHttpRequest。 使用 Ajax 技术网页应用能够快速地将增量更新呈现在用户界面上，而不需要重载（刷新）整个页面，这使得程序能够更快地回应用户的操作。利用它，我们可以实现免刷新上传文件的操作，可以高效的运用在词云的生成过程。

jQuery 是一个快速、简洁的 JavaScript 框架，，设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互。

```
$('#file').change(function (e) {}) #这是 HTML 的 Scirpt 标签中插入上传文件的脚本

@app.route('/upload_file', methods=['POST', 'GET'])      #这是获取文件流的操作
def upload():
    if request.method == 'POST':
        f = request.files['file']
```

2.2.5.API 接入

API 的接入有利于提升网页的吸引力。高德地图 API 无疑符合网页的制作需求。例如，我们导入了高德官方提供的柱状图 Script。核心 script 如下：

```
    pl.setStyle({
  clickInfo.setMap(map);
  clickInfo.hide();
  // 动画测试
  map.on('click', function (e) {
  });
```

在此 API 中，我们主要引用了 Loca API 2.0，它采用了和其 1.3 版本中不同的架构模式和渲染管线，极大的提升了性能和渲染效果。我们在 2.0 版本中也引入了自定义镜头动画、图层动效、光照和材质等能力。

同时 API 对数据源进行了标准化，新版本中仅支持标准的 GeoJSON 格式数据。

2.2.6.词云

词云的实现流程很简单，即导入目标文件并输出文件流即可，代码实现如下。

```
    def makeWordcloud(image_name):      #image_name 存放 image_name 的路径字符串
    con = sqlite3.connect('data.db')
    cur = con.cursor()
    sql = 'select info from movie250 '
    #----------省略的代码————————#
        plt.axis('off')
    plt.savefig('output.jpg', dpi=1600)
```

3. 系统测试

3.1. 测试日志：

在 dev1.0 版本的测试中我们是只是有一个普通的后台来对数据进行爬取，并只能将爬取过的豆瓣电影信息存入到 Excel 表格中，这样对于用户来说没有一个友好的前端页面来对后端进行数据的交互，考虑到我们是做的是一个面向广大用户的一个爬取电影信息的一个平台，我们就开始学习一些基础的 Web 前端的学习，并且在学习的过程中学会使用 Html、CSS、JS 来对我们的这个平台进行页面的设计以及进一步的优化这个平台页面。

在 dev1.1 版本的测试中我们发现只仅有 Excel 来对爬取的数据进行储存并进行对数据的展示，这样显示的数据比较的单一，并且我们想统计一些关于电影相关的数据以及关键词合成词云就需要使用数据库来进行操作。

在 dev1.2 版本测试中，我们发现对于数据的爬取速度太慢，我们将单线程优化为多线程并发爬取数据，这样极大的优化了对于数据的爬取速率。单线程爬取资源这种方式会导致爬取速度十分缓慢，造成用户不好的使用体验，基于此问题我们改进优化代码，实现了对网络信息的多线程爬取，大大提高了信息获取速度以及用户使用体验。

3.2 演示系统截图

（1）网站首页：

图 3.2.1 网站首页

（2）影视信息部分：

图 3.2.2 影视信息部分

（3）爬取 top250 电影后的页面显示：
（页面中包含电影信息以及下载 excel、更新数据库按钮）

图 3.2.3 爬取 top250 电影后的页面显示

（4）城市院线电影信息搜索界面：

（包括搜索栏以及高德地图院线电影上映柱状分布图）

图 3.2.4 城市院线电影信息搜索界面

（5）以下是以北京为例搜索到的院线电影上映信息：

图 3.2.5 院线电影上映信息

（6）豆瓣电影 top250 评分分布图：

图 3.2.6 豆瓣电影 top250 评分分布图

（7）词频统计界面：（允许用户自主上传图片生成相应图云并支持下载以作他用）

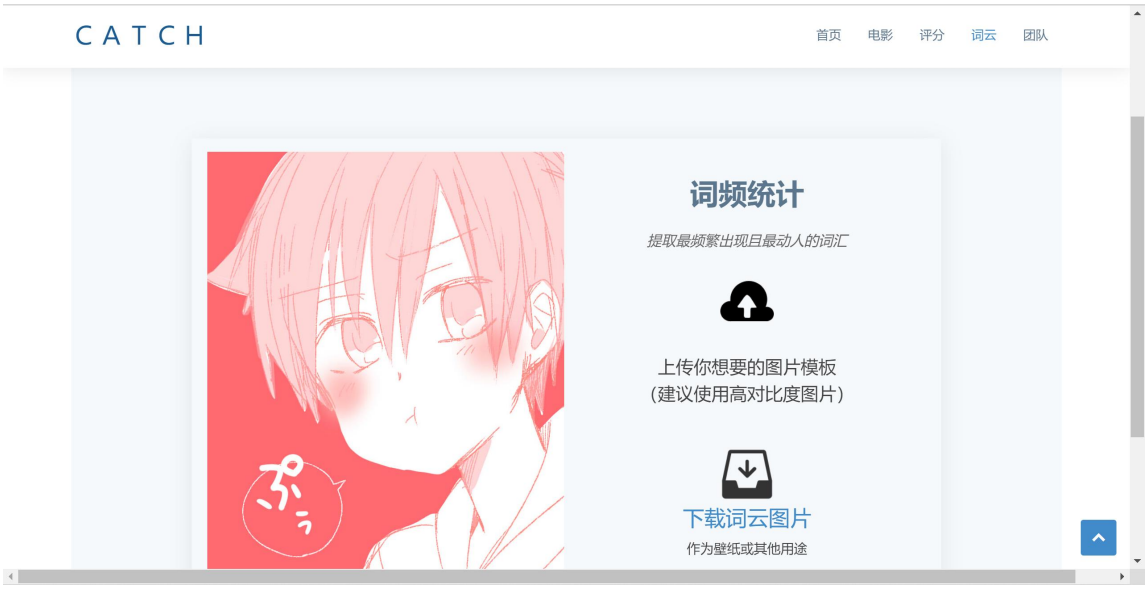

图 3.2.7 词频统计界面

（6）生成词云样例如下：

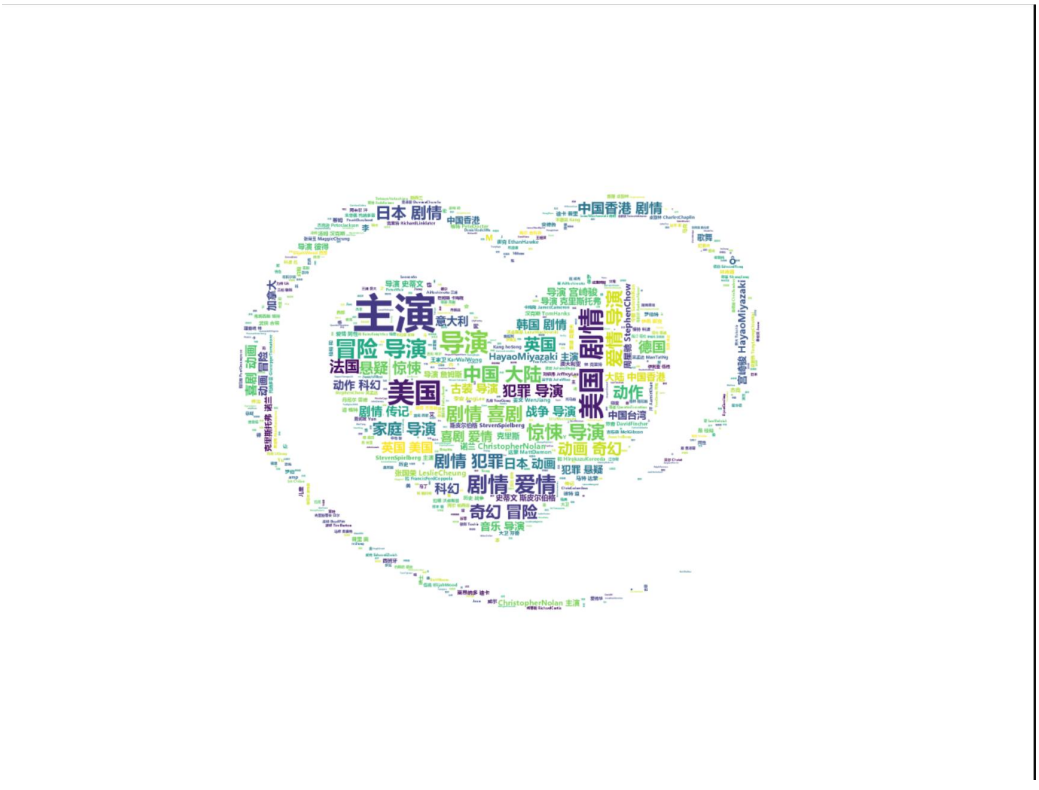

图 3.2.7 生成词云样例

（7）团队信息展示页面：

图 3.2.8 团队信息展示页面

4. 总结

4.1. 系统特点：

（1）多线程爬取网络数据
（2）数据可视化处理
（3）高德 API 展示数据统计结果
（4）数据库编程实现信息存储

4.2. 开发过程中遇到的问题：

（1）我们是只是有一个普通的后台来对数据进行爬取，并只能将爬取过的豆瓣电影信息存入到 Excel 表格中，这样对于用户来说没有一个友好的前端页面来对后端进行数据的交互，考虑到我们是做的是一个面向广大用户的一个爬取电影信息的一个平台，我们就开始学习一些基础的 Web 前端的学习，并且在学习的过程中学会使用 Html、CSS、JS 来对我们的这个平台进行页面的设计以及进一步的优化这个平台页面。

（2）我们发现只仅有 Excel 来对爬取的数据进行储存并进行对数据的展示，这样显示的数据比较的单一，并且我们想统计一些关于电影相关的数据以及关键词合成词云就需要使用数据库来进行操作。

（3）我们发现对于数据的爬取速度太慢，我们将单线程优化为多线程并发爬取数据，这样极大的优化了对于数据的爬取。单线程爬取资源，这种方式会导致爬取速度十分缓慢，

造成用户不好的使用体验，基于此问题我们改进优化代码，实现了对网络信息的多线程爬取，大大提高了信息获取速度以及用户使用体验。

4.3.收获总结：

通过小组大作业的学习，我们对于 python 本门课程有了更深一步的理解。我们本门课程课时不算很长，所以通过小组作业能学习更多 Python 的知识并将之应用到我们的实践中去，爬虫知识的学习也丰富了我们对网站信息的理解，同样本次大作业涉及知识面很广，这也是我们在开发生涯中的处女之作。我们明白了整个开发流程，以及明白了团队开发的重要性。学会了后端怎么和数据库进行交互，用数据库存储信息的时候我们应用到了用数据库的 sql 语句对爬取的数据进行查询以及统计，这要求我们学习巩固数据库相关知识并学习基于 sqlite 库对数据库信息的一些基本操作。再者，我们还在开发中运用了课上所学的词云加以应用，在我们爬取的电影资源中生成高频词汇词云并允许用户下载。以及运用了在课堂上学会的多线程的知识，另外还学习了相关前端知识能够进行设计符合我们平台的交互页面，数据可视化处理能够向用户呈现出更良好的数据统计以及分析。

5.参考文献

（1）IT 私塾，Python 爬虫编程基础 5 天速成（2021 全新合集）Python 入门+数据分析[EB/OL]，https://www.bilibili.com/video/BV12E411A7ZQ?spm_id_from=333.337.search-card.all.click，2020.03.21/2022.04

（2）张瑞霞，Python 开发技术 2022 春季 2122110[EB/OL]，https://www.educoder.net/classrooms/QXHBZOAS/video，2022.03/2022.04

（3）梦凝哲雪，获取中文 URL 参数 显示乱码[EB/OL]，https://blog.csdn.net/Klhz555/article/details/107861401，2020.12.07/2022.04

（4）高德开放平台，概述-地图 JS API v2.0|高德地图 API[EB/OL]，https://lbs.amap.com/api/jsapi-v2/summary/ ，2022.03.09/2022.04

（5）Python 研究者，基于 Flask 开发网站 -- 前端 Ajax 异步上传文件到后台（文末送书）[EB/OL]，https://cloud.tencent.com/developer/article/1861234，2021.08.13/2022.04