爬虫与数据可视化作业报告

|  |  |
| --- | --- |
| 系别 | 软件工程 |
| 学生 | 朱金龙 |
| 学号 | 201992291 |
| 班级 | 软测1905 |
| 成绩 |  |

2021 年 12 月 28 日

1. **项目背景**

**本次大作业中爬虫所选url为csdn中开源社区中的Java项目网页（**[项目 · 探索 · GIT CODE](https://gitcode.net/explore/topics/java)**），针对每个项目的开发者，项目名称，获得的star,fork ，请求合并，issue，项目所使用的开源协议，以及项目简介进行数据爬取，并存储到mongodb数据库中。在数据分析阶段使用pandas的分析方法，来进行数据的计算，最终使用pyecharts来将分析后数据做成可视化图表。**

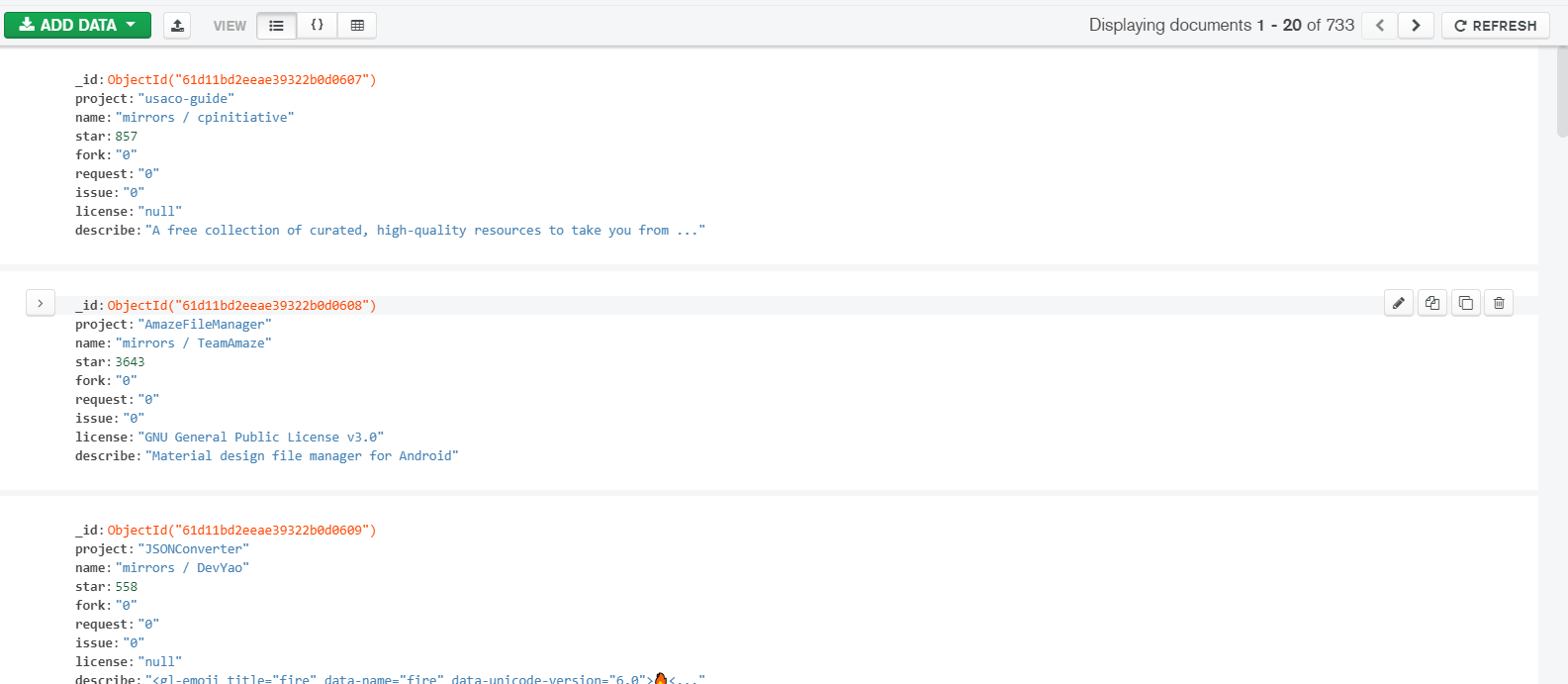
1. 数据采集模块

本部分中主要使用了re，requests 2个模块的功能，其中用requests的get方法来获取网页的html文件，并通过分析网页的内容与结构，来用re表达式进行正则提取

代码部分如下

import requests  
import re  
import pymongo  
def Gethtml(url):  
 headers = {  
 "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.55 Safari/537.36 Edg/96.0.1054.43"  
 }  
 html = requests.get(url, headers=headers)  
 return html.text  
  
  
def issue\_count(content):  
 list = []  
 issue\_count = re.compile(r'<a class="d-none d-xl-flex align-items-center icon-wrapper issues has-tooltip".\*?Issue.\*?</use></svg>\n(?P<issue>.\*?)\n</a>',re.S)  
 result = issue\_count.finditer(content)  
 for item in result:  
 list.append(item.group("issue"))  
 return list  
  
  
def lisence\_kind(content):  
 list = []  
 lisence = re.compile(r'<a class="d-none d-xl-flex align-items-center icon-wrapper issues has-tooltip".\*?许可证.\*?</use></svg>\n(?P<lisence>.\*?)\n</a>',re.S)  
 result = lisence.finditer(content)  
 for item in result:  
 list.append(item.group("lisence"))  
 return list  
  
  
def request\_count(content):  
 list = []  
 request\_count = re.compile(r'<a class="d-none d-xl-flex align-items-center icon-wrapper merge-requests has-tooltip".\*?</use></svg>\n(?P<request>.\*?)\n</a>',re.S)  
 result = request\_count.finditer(content)  
 for item in result:  
 list.append(item.group("request"))  
 return list  
  
  
def fork\_count(content):  
 list = []  
 fork\_count = re.compile(r'<a class="align-items-center icon-wrapper forks has-tooltip".\*?</use></svg>\n(?P<fork>.\*?)\n</a>',re.S)  
 result = fork\_count.finditer(content)  
 for item in result:  
 list.append(item.group("fork"))  
 return list  
  
  
def star\_count(content):  
 list = []  
 re\_count = re.compile(r'<span class=".\*?-star-count">\n(?P<count>.\*?)\n</span>')  
 result = re\_count.finditer(content)  
 for item in result:  
 list.append(item.group("count"))  
 return list  
  
  
def Name(content):  
 list = []  
 re\_name = re.compile(r'<span class="namespace-name">\n(?P<name>.\*?)\n')  
 result = re\_name.finditer(content)  
 for item in result:  
 list.append(item.group("name"))  
 return list  
  
  
def Project(content):  
 list = []  
 re\_project = re.compile(r'<span class="project-name">(?P<project>.\*?)</span>')  
 result = re\_project.finditer(content)  
 for item in result:  
 list.append(item.group("project"))  
 return list  
  
  
def Describe(content):  
 list = []  
 re\_describe = re.compile(r'<p data-sourcepos="1..\*?" dir="auto">(?P<describe>.\*?)</p>')  
 result = re\_describe.finditer(content)  
 for item in result:  
 list.append(item.group("describe"))  
 return list  
  
  
def Unit(html):  
 list = []  
 li = re.compile(r'<li class="d-flex .\*?project-row">(?P<unit>.\*?)</li>',re.S)  
 result = li.finditer(html)  
 for item in result:  
 list.append(item.group("unit"))  
 return list  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 myclient = pymongo.MongoClient('mongodb://localhost:27017/')  
 mydb = myclient['lab1']  
 mycol = mydb['information']  
 k = 0  
 list = []  
 lists = [[]for i in range(750)]  
 #第一步获取一个页面的html,第二步获取一个页面中的每个项目所在的函数<li class="d-flex project-row">的代码段，在代码段中进行匹配，若没有则记为空  
 #建立循环，获取50页html的内容  
 for i in range(37):  
 #每一页的url,以i为变化  
 url = f'https://gitcode.net/explore/topics/java?page={i+1}'  
 #获取html  
 text = Gethtml(url)  
 #获取每个html中的项目的信息盒子  
 unit = Unit(text)  
 #输出项目数量  
 #建立循环，将每个页面中的每个项目的信息进行提取  
 for j in unit:  
 #对各项信息进行提取，共724个项目的信息。  
 project = Project(j)  
 name = Name(j)  
 star = star\_count(j)  
 #去除数据中的其他符号  
 star[0] = star[0].replace(',','')  
 fork = fork\_count(j)  
 request = request\_count(j)  
 issue = issue\_count(j)  
 lisence = lisence\_kind(j)  
 if len(lisence) == 0:  
 lisence.append('null')  
 describe = Describe(j)  
 if len(describe) == 0:  
 describe.append('null')  
 star[0] = int(star[0] )  
 mycol.insert\_one({"project":project[0],"name":name[0],"star":star[0],"fork":fork[0],"request":request[0],"issue":issue[0],"license":lisence[0],"describe":describe[0]})

运行结果：此代码最终将会把所有爬取到的信息，存储到本地的mongodb中，并会自行生成名为lab1的数据库和information的集合。截图如下所示：



**2．**数据分析与展示模块

在数据分析中，我们使用了pandas,通过访问之前存储数据的数据集合，将集合中的数据转成dateframe格式，进而对数据中我们需要分析的列进行groupby操作，再将分析后的数据转换成list格式，传入pyecharts中，生成图表文件

代码部分如下：

import pandas as pd  
from pyecharts import Bar  
from pyecharts import Pie  
import pymongo  
import numpy as np  
#把数据库中的数据读取出来,转换成dataframe格式  
myclient = pymongo.MongoClient('mongodb://localhost:27017/')  
mydb = myclient['lab1']  
mycol = mydb['information']  
data = pd.DataFrame(list(mycol.find()))  
#提取行和列,这是把所有的的行中的name,project.... 提取了出来（去除mongoDB中的id项）  
data = data.loc[:,['project','name','star','fork','request','issue','license','describe']]  
#查询各种不同的license在项目中的占比,提取出不同许可证的数量转换成字典模式  
agg\_find\_license = data.groupby('license').agg({'license':'count'})  
license\_dict = agg\_find\_license.to\_dict('dict')  
license\_data = license\_dict['license']  
#删除null项目  
del license\_data['null']  
#提取出license\_data中的键和值  
license\_list = list(license\_data.keys())  
count = []  
for item in license\_data:  
 count.append(license\_data[item])  
#绘制饼图  
pie = Pie('项目中许可证占比',height=1000,width=1200)  
pie.add('',license\_list,count,is\_label\_show=True)  
pie.render('饼图.html')  
#pandas 对star。。。。类型转换  
data[['star']] = data[['star']].apply(pd.to\_numeric)  
data[['fork']] = data[['fork']].apply(pd.to\_numeric)  
data[['request']] = data[['request']].apply(pd.to\_numeric)  
data[['issue']] = data[['issue']].apply(pd.to\_numeric)  
#找出star数值排在前十的作者  
agg\_find\_name = data.groupby('name').agg({'star':'sum','fork':'sum','request':'sum','issue':'sum'})  
agg\_find\_name = agg\_find\_name.sort\_values(by=['star'],ascending=False)  
dict1 = agg\_find\_name.to\_dict('dict')  
star\_list = []  
fork\_list = []  
request\_list =[]  
issue\_list = []  
star\_dict = dict1['star']  
for item in star\_dict:  
 star\_list.append(star\_dict[item])  
fork\_dict = dict1['fork']  
for item in fork\_dict:  
 fork\_list.append(fork\_dict[item])  
request\_dict = dict1['request']  
for item in request\_dict:  
 request\_list.append(request\_dict[item])  
issue\_dict = dict1['issue']  
for item in issue\_dict:  
 issue\_list.append(issue\_dict[item])  
name0 = list(star\_dict.keys())  
name\_list = []  
count = 0  
for i in name0:  
 if 'mirror' in i:  
 i = i.split(' / ')[1]  
 name\_list.append(i)  
 count = count+1  
#数据分析完成，图表可视化，1.柱状图显示排行前十的项目的项目名，star,fork,的数值 2.饼状图显示各个许可证在项目总数中的占比  
#pyecharts,columns中的值是array2  
bar = Bar('gitcode上各类数据总计排行前十大佬',title\_pos='left',height=1000,width=1200)  
bar.add("star", name\_list[0:10], star\_list[0:10], mark\_line=["average"], mark\_point=["max", "min"])  
bar.add("fork", name\_list[0:10], fork\_list[0:10], mark\_line=["average"], mark\_point=["max", "min"])  
#bar.add("合并请求", columns, data3, mark\_line=["average"], mark\_point=["max", "min"])  
#bar.add("issue", columns, data4, mark\_line=["average"], mark\_point=["max", "min"])  
bar.render('柱状图.html')

运行截图如下，本阶段会在执行后在项目文件夹中生成名为“柱状图.html”和”饼图.html”的文件，图表分别是统计了项目中获得star数目排行前十的作者，并展示了他们的star总数和fork数，饼图则是表示了在所有项目中，大家所采用的开源协议的种类和使用占比。

运行截图如下：

1. 