大数据技术与应用课程报告

——2018 级

课题名称： 基于 Flask 的拉勾网的数据可视化

姓　名 ____肖龙_____

学　号 ____180620130___

班　级 ____大数据 1801__

成　绩 _____

2020 年 12 月 07

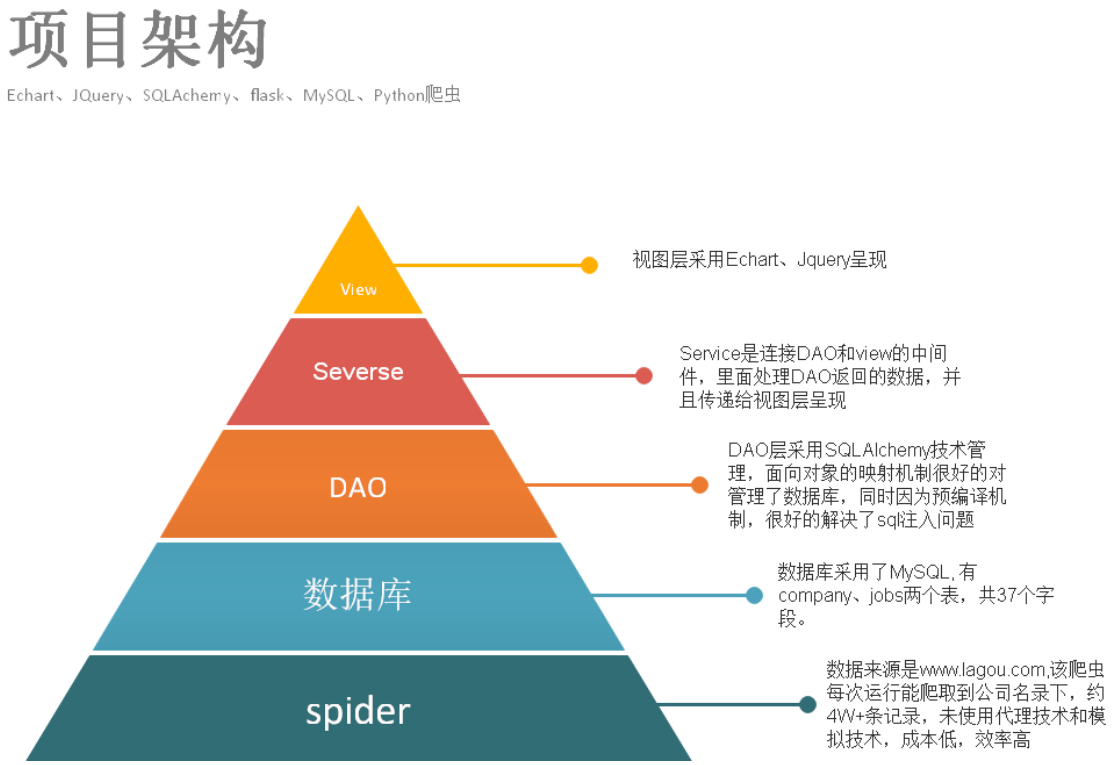
大数据技术分析与应用报告

作者：肖龙

时间：2020-10-25

架构

概述



技术栈

`JQuery`：对数据进行展示和前端网页需要该项技能，能更加方便和优雅的写 JavaScript 代码，作为一个前端框架简化了开发流程。

`AJAX`: 能以较小的代价（更低的流量）来进行数据的展示

`JavaScript、CSS、HTML`：用来展示前端网页

python 及其 reuqests 库： 在本例中，用于完成爬虫代码

Echart: 展示图片数据，绑定一些事件对数据进行深度显示

难点

- BS 交互，flask 充当了服务器，解决了这个问题
- 浏览器中展示图片，Echart 具备强大的扩展库和大量的实例，根据上面的实例完全可以快速的部署上自己的项目
- MySQL 对象管理，使用 SQLAlchemy 解决，具有优秀的对象映射机制、完美的融合 SQL 语法
- 爬虫使用 AJAX 异步加载：对请求的数据接口进行分析，深入了解 lagou.com 参数运行机制后，采用 requests 库模拟请求，然后使用 SQLAlchemy 将爬下来的数据存入数据库中。

该 demo 优点

- 整套流程已经能够完全跑通
- 完全自动的更新数据和存入数据，数据获取的非常稳定。并且因为直接向接口请求数据，数据获取的速度快，获取的量也大
- 项目的分离程度大 DAO、Service、spider 各司其职，便于维护

缺点

- 小，展示页面种类较少
- 爬虫过程中，出现的一个接口的最大的请求页数为 300 页，如果要求请其他的页面，要变化参数，但是变换参数后的数据可能前面已经爬取，因此不好处理

项目目录结构

```
(venv) D:\code\python\hello>tree
文件夹 PATH 列表
卷序列号为 B617-B3E0
D:.
├.idea
|   ├dataSources
|   |   └febf9f3a-0c3f-4d55-bb47-55efced79197
|   |       └storage_v2
|   |           └_src_
|   |               └schema
|   └inspectionProfiles
├bean
|   └__pycache__
├Dao
|   └__pycache__
├login
|   └__pycache__
├myError
|   └__pycache__
├save
├service
|   └__pycache__
├spider
|   └data
|       ├100556
├static
|   ├bmap
|   ├display
|   ├echarts
|   |   ├build
|   |   ├extension
```

image-20201025220735420

具体功能

数据库

有两个表，分别是 company 和 jobs，前者存储公司数据，后者存储职位数据

- company 表

```
create table if not exist company
(
    company_id             int          not null
        primary key,
    company_full_name      varchar(100) not null,
    company_short_name     varchar(50)  null,
    company_logo           varchar(200) null,
    city                   varchar(20)  not null,
    industry_field         varchar(100) null,
    company_features       varchar(200) null,
    finance_stage          varchar(8)   null,
    company_size           varchar(15)  null,
    interview_remark_num   int          null,
    position_num           int          null,
    update_time            date         null,
    process_rate           int          null,
    approve                int          null,
    company_combine_score  float        null,
    is_has_valid_position  tinyint(1)   null,
    other_label            varchar(400) null,
    match_score            float        null
);
```

- jobs 表·

```
-- auto-generated definition
create table jobs
(
    company_id         int                      not null,
    position_id        int                      not null
        primary key,
    job_nature         varchar(5)               not null,
    finance_stage      varchar(50)              not null,
    company_name       varchar(20)              not null,
    company_full_name  varchar(40)              not null,
    company_size       varchar(20)              not null,
    industry_field     varchar(50)              not null,
    position_name      varchar(80)              not null,
    city               varchar(10)              not null,
    create_time        date                     null,
    salary             varchar(20)              not null,
    work_year          varchar(10) default '不限' null,
```

```
    education          varchar(10)             not null,
    position_advantage varchar(100)            null,
    company_label_list varchar(400)            null,
    user_id            varchar(10)             not null,
    company_logo       varchar(100)            null,
    district           varchar(20)             not null
    -- 可以创建一个外键约束，将 job detail 数据放在这个表中，这样在 SQLAlchemy 中对应的对象中创建一个对应的对象即可
);
```

对应的 SQLAlchemy 中管理的对象为

```
from sqlalchemy import Column, String, Float, Date, Integer
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()


class Company(Base):

    __tablename__ = "company"

    company_id = Column(Integer, primary_key=True)
    company_full_name = Column(String(100), nullable=False)
    company_short_name = Column(String(50))
    company_logo = Column(String(200))
    city = Column(String(20), nullable=False)
    industry_field = Column(String(100))
    company_features = Column(String(200))
    finance_stage = Column(String(8))
    company_size = Column(String(15))
    interview_remark_num = Column(Integer)
    position_num = Column(Integer)
    update_time = Column(Date)
    process_rate = Column(Integer)
    approve = Column(Integer)
    company_combine_score = Column(Float)
    is_has_valid_position = Column(Integer)
    other_label = Column(String(200))
    match_score = Column(Float)


class Jobs(Base):

    __tablename__ = "jobs"

    company_id = Column(Integer, primary_key=True)
    position_id = Column(String(10), nullable=False)
```

```
    job_nature = Column(String(5), nullable=False)
    finance_stage = Column(String(50), nullable=False)
    company_name = Column(String(20), nullable=False)
    company_full_name = Column(String(40), nullable=False)
    company_size = Column(String(20), nullable=False)
    industry_field = Column(String(50), nullable=False)
    position_name = Column(String(80), nullable=False)
    city = Column(String(10), nullable=False)
    create_time = Column(Date)
    salary = Column(String(20), nullable=False)
    work_year = Column(String(10), nullable=False, default='不限')
    education = Column(String(10), nullable=False)
    position_advantage = Column(String(100))
    company_label_list = Column(String(400))
    user_id = Column(String(10), nullable=False)
    company_logo = Column(String(100))
    district = Column(String(20), nullable=False, default='北京')
```

就这样完成了数据库的创建

爬虫

分三步

1. 得到公司数目和名称
    1. 公司名称部分，接口下的数据进行请求，设置 position:‘全国',发起请求共获得 62 页共 624 个公司类目
    2. 将数据存入数据库
2. 得到公司的具体职位
    1. 对第一步得到公司爬取其职位数据，获得其 pisition_id 信息
    2. 将获得的数据存入数据库中
3. 得到职位的具体信息
    1. 使用上一步取得的 position_id 属性，带上 cookie 请求网址 https://www.lagou.com/jobs/**{position_id}**.html,`
    2. 将对应的数据存入数据库中

DAO

主要有四个方法，分别是

getCompanyNum: 得到公司的数目

`getCompanyCityData`： 得到城市中公司的数据

`getBar`：为创建 bar 产生数据

`threeParm`： 对参数进行处理并且返回数据

`getDBSession` 得到数据库连接

myError

定义了几个类型测试，当数据不满足输入要求时，抛出异常

service

处理数据，返回给将数据发送给 flask，将数据返回给 flask

`deal`: 处理函数，处理给 `summit` 处提交过来的数据，并进行一个模糊的判断该特征是不是在符合查询到的数据，如果符合，就保留下来，不符合就删除

`dealThreeParm`：辅助处理的函数

`dealData`：辅助处理函数

`dealGetBar`： 处理 bar 图形的数据，将数据返回给 Flask，同时 map 也调用的是这个函数，对数据处理。

`dealGetBarData`： 处理 bar 查询到的数据

`getCityData`： 按照城市数据和公司数据随机返回 20 个数据，将数据返回给 flask

static

放置了静态文件，这些文件需要文档出现的时候加载出来

templates

放置了一些模板

app.py

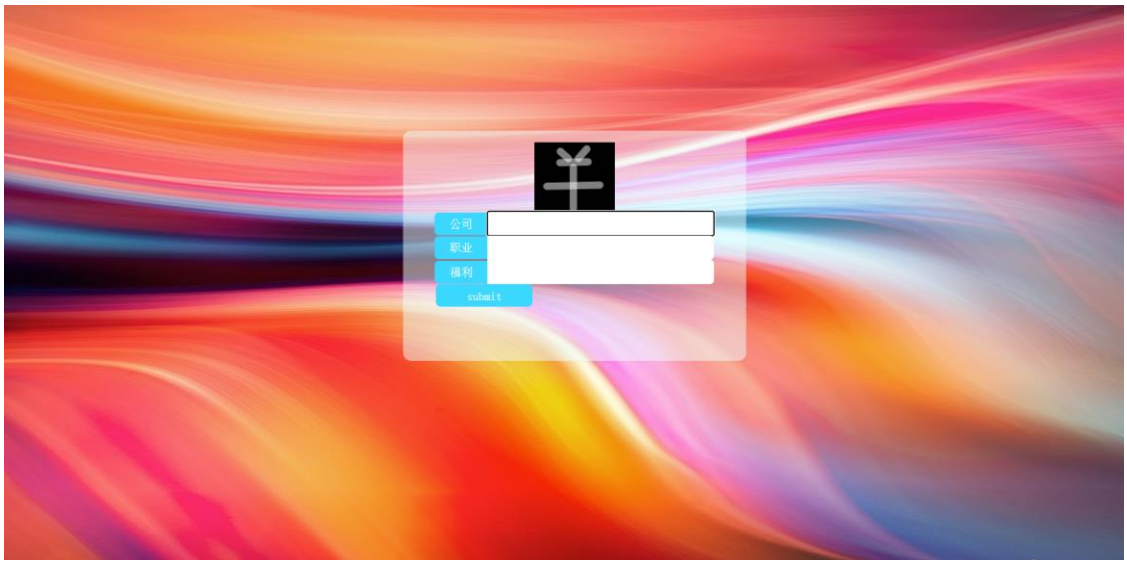
- 监控了三个页面分别是
  - `/`： 返回 index
  - `/submit`： 处理 submit 的数据
  - `/bar_chart`： 处理三个 `ajax` 请求, 处理参数后分别返回对应数据，然后前端进行展示

实现效果

可登录网站：http://xiaolong.vaiwan.com 或者 https://xiaolong.vaiwan.com 查看 demo 效果

首页

进入该网站后，首先应出现的是网站的索引页面如下图

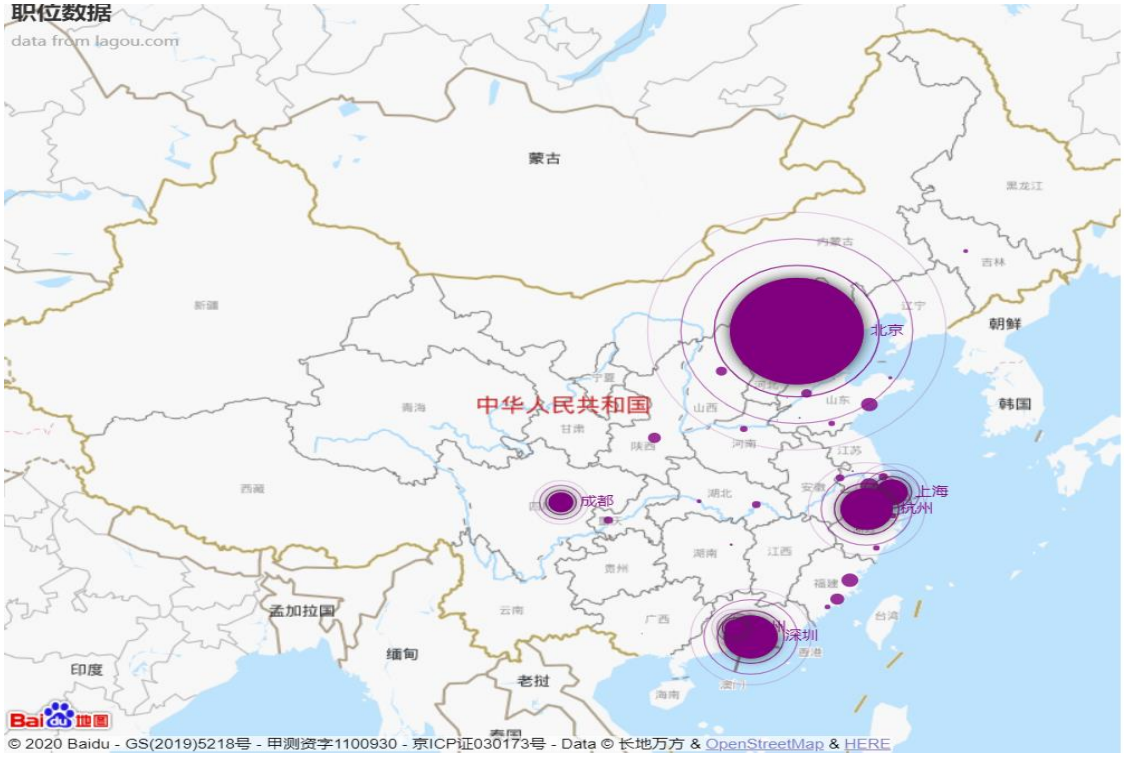


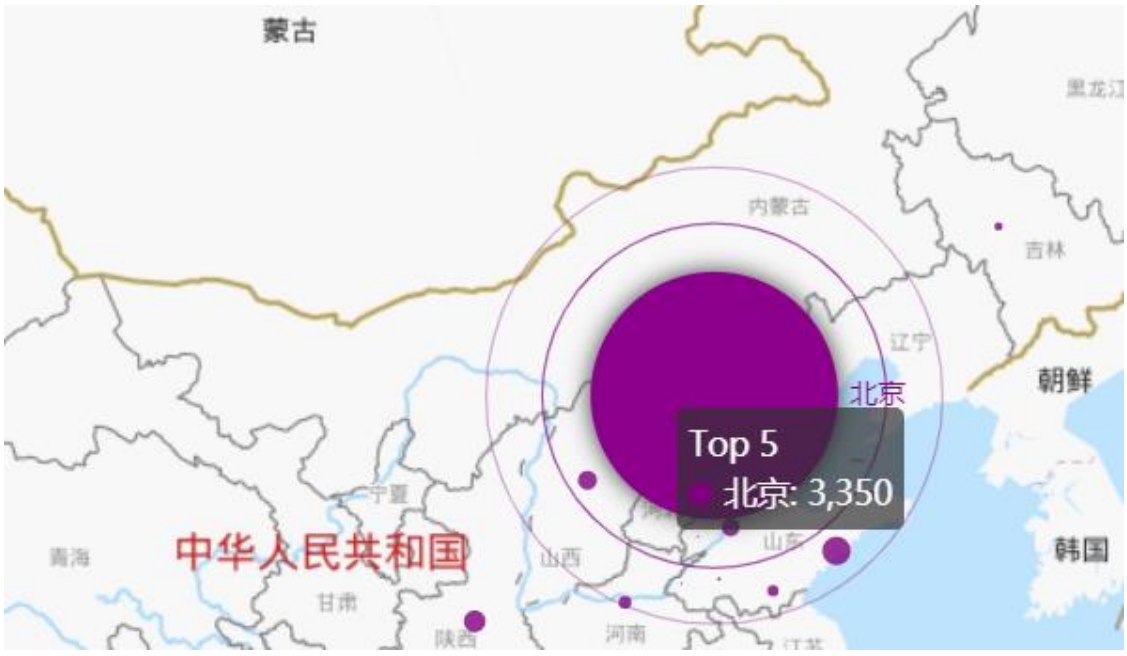
公司一列可以填入一个公司的公司名称。例如字节跳动，美团，如果没有找到这个公司默认显示字节跳动。职业和福利一栏支持模糊搜索，只要这个出现任意一个字符，那么都会返回对应的信息，例如输入如以下图片

会返回一个页面，该页面包括三个部分，分别是作为大地图的第一部分

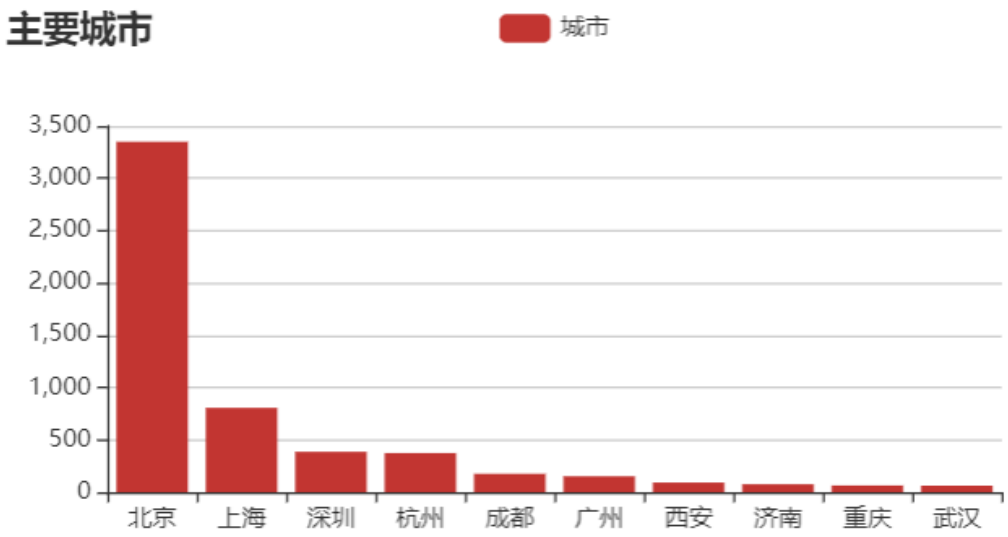
功能：

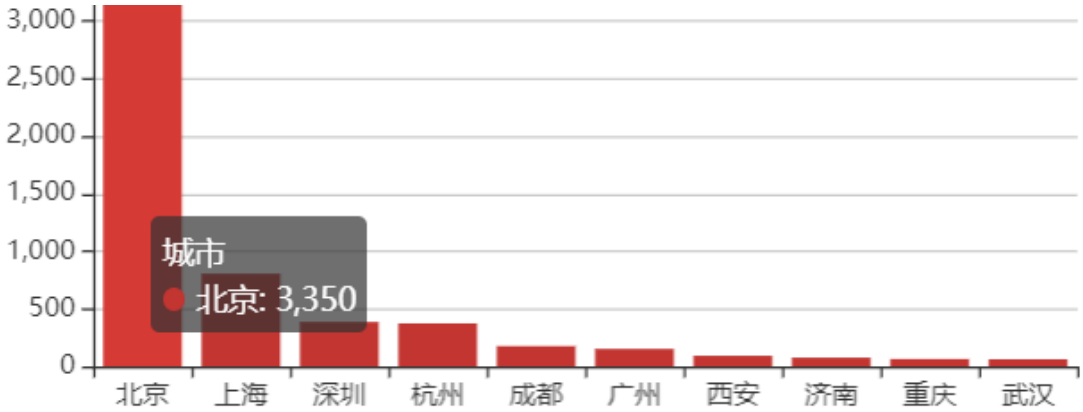
- 职位地址的数据显示，对于不同的职位，将职位的信息标注在地图上
- 鼠标移入显示具体的职位数字





柱状体部分，可以显示前 10 的具体城市有哪些，点击之后，在第三部分显示具体有哪些职位，如果对这个职位感兴趣，可以进入到报名页面报名





广告系统资深后端高级开发工程师	全职C轮25k-50k3-5年本科
后端资深研发工程师-商业变现方向	全职C轮25k-50k3-5年本科
品牌广告后端资深研发工程师	全职C轮25k-50k3-5年本科
EDA 工程师	全职C轮20k-40k不限 本科
大数据研发负责人	全职C轮50k-80k不限 本科
搜索算法工程师 - 电商	全职C轮30k-50k3-5年本科
广告技术数据科学家	全职C轮30k-60k3-5年本科

这就是这个 demo 的全部内容，体验可以访问

http://xiaolong.vaiwan.com/

所有的代码在

HTTP:https://github.com/low-bee/flask-web-lagou.git

SSH: git@github.com:low-bee/flask-web-lagou.git

ps:没有任何防御措施，网页使用内网穿透，可能会有点卡。