《软件实践》课程实验报告

暑期学校实验项目：高考志愿填报助手

小组名称	知识推理及图谱可视化						
姓　　名	高成睿	专业	人工智能	班级	091181	学号	09118136
实验时间	2020.8.31-2020.9.23		指导教师	孔祥龙		成绩	

一、实验背景和目的

实验背景:

1. 大二学年学习了 python 语言，数据库概论等课程，使用 python 进行网站开发是对之前课程很好的回顾，能加深对前置课程的理解。
2. 大三学年开始学习知识推理与表达，该实践项目加入了知识图谱的相关知识，为后续课程做了铺垫。

该实验在课程设置中起到了承前启后的作用，通过实践，既增强了学生对 python 语言的掌握，也加深了学生对人工智能相关课程的理解。

目的：

1. 增强 python 的使用能力，通过 gitee 平台使学生熟悉团队协作的开发方式，了解 git 的使用方法。同时，也使学生了解网站开发的流程。
2. 与 AI 知识相链接，在实践中巩固专业知识。

二、小组任务和个人任务

小组任务：

1、 依据高校分数线进行统计分析，生成相应图表，展示变化趋势；
2、根据用户输入的动态个人信息进行相应的可视化；
3、以地图为依据生成可视化界面，展示高校信息；
4、基于知识图谱的推理和问答；
5、对应的 django 框架编写

个人任务：组长分配的任务为对接数据库，提供从数据库中提取数据并转换为所需数据结构的函数。实际开发中需要写的提取数据库数据的函数不多，所以自行设计实现了大学录取分数查询，专业录取分数查询以及考生邻近省份一流高校查询的模块。前两个模块可以方便用户查看某个高校或专业录取分数和位次的走势，第三个模块可以提供邻近省份的名校名单，帮助考生进行学校选择。

三、个人任务需求分析

需求 1（组长分配）：数据库使用

a. 接收组内其他成员的数据需求，了解所需的数据和数据结构。
b. 使用数据库，从数据库中取数据。
c. 将取出的数据转化为所需数据结构，作为返回值。
d. 提供函数接口，方便组内其他成员的开发，统一并规范小组内数据的使用。

需求 2（自行发挥）：大学及专业录取分数查询

a. 实现从数据库中读取某大学在某省份某科类三年间（2017-2019）的录取分数线和录取位次，将其存储在列表中。封装为函数，返回值为存有数据的列表。函数的最后一个形参为专业，其默认值为 all，即大学的投档线，从而同时实现对大学录取分和

专业录取分的查询。

b. 根据存有三年录取分数和位次的列表，绘制分数的折线图和位次的折线图。

c. 编写 html，从前端接收用户输入的大学名称，省份名称，科类名称以及专业名称（模糊查询）。后端按用户输入进行查询，生成图表，返回到前端显示。

需求 3（自行发挥）：考生邻近省份的一流高校查询

a. 根据中国地图，设计一个字典表示邻近省份。字典中的键为当前省份的 ID，值为存有所有邻近省份 ID 的列表。邻近的标准为两个省份在地图上相邻。

b. 实现从给定考生省份，查询考生所在省份及其邻近省份的双一流学校。用字典存储，键为省份名称，值为高校名称构成的列表。

c. 编写 html，从前端接收用户输入的省份名称。后端根据用户输入查询出邻近省份的高校字典，传到前端页面进行显示。

四、实验过程（需附上关键代码及相关说明）

1. url.py 的 url 配置

```
urlpatterns=[path('CollegeInfoGraphs',views.CollegeInfoGraphs,name='CollegeInfoGraphs'),
    path('MajorInfoGraphs',views.MajorInfoGraphs,name='MajorInfoGraphs'),
    path('TopUnivOfNeighbors',views.DisplayTopUnivOfNeighbors,name='TopUnivOfNeighbors'),]
```

这三个 url 对应的功能分别为为显示某学校近三年录取分数和位次，显示某学校某专业近三年录取分数和位次和查询考生邻近省份的一流高校。在 view.py 中有分别的函数与之对应。

2. views.py 中的函数

a. 函数 getMajorScoresRanking(pID,cID,y,mName)

```
#Get certain major scores in different univ.
def getMajorScoresRanking(pID,cID,y,mName):
    #参数为省份 ID（整数 1-34），科类 ID（整数 1-3），年份（整数）和专业名称
    majorList=Majors.objects.filter(provinceID=pID,
                                    categoryID=cID,
                                    year=y,
                                    majorName=mName)
    scoresDict={}
    for major in majorList:
        scoresDict[major.collegeID.collegeName]=major.minScore
    scoresOrder=dict(sorted(scoresDict.items(), key = lambda kv:kv[1],reverse=True))
    return scoresOrder
```

输入省份 ID，科类 ID，年份和专业名称，使用 Django 中 models 提供的 filter 方法，在数据库中查询出与形参相符的数据，类型为 querySet，即一个有 major 对象的列表。将 querySet 中的数据转为字典，键为 collegeName（这里使用外键 collegeID 表示了 college 对象），值为专业最低分。

使用 sorted()函数排序，返回按分数降序排列的字典。

该函数是为组内其它成员提供的数据提取和处理函数。

b. 统计 985，211 和双一流学校的函数

```
#Get the number of 985,211 and top in every province
def get_data_985():
    list_985=[]
    for i_ in range(1,35):
        count_985=Colleges.objects.filter(provinceID=i_,project985=True).aggregate(Count('collegeID'))
        list_985.append(count_985['collegeID__count'])
    return list_985
def get_data_211():
    list_211=[]
    for i_ in range(1,35):
        count_211=Colleges.objects.filter(provinceID=i_,project211=True).aggregate(Count('collegeID'))
        list_211.append(count_211['collegeID__count'])
    return list_211


def get_data_top():
    list_top=[]
    for i_ in range(1,35):
        count_top=Colleges.objects.filter(provinceID=i_,top=True).aggregate(Count('collegeID'))
        list_top.append(count_top['collegeID__count'])
    return list_top
```

上面三个函数，分别统计了各个省份的 985，211 和双一流学校的数量。遍历省份 ID，使用 filter 方法查询出各个省份的 985，211 和双一流学校，类型为 querySet。再使用 querySet 的 aggregate 方法按 collegeID 进行计数，然后将计数结果存入列表 list_985/list_211/list_top。将列表 list_985/list_211/list_top 返回。

```
def get_data():
```

```
    series=[]
    list_985=get_data_985()
    list_211=get_data_211()
    list_top=get_data_top()
    for i_ in range(0,34):
        temp={}
        province=Provinces.objects.filter(provinceID=i_+1)
        temp["name"]=province[0].provinceName
        temp["value"]=i_
        temp["project985"]=list_985[i_]
        temp["project211"]=list_211[i_]
        temp["doubleTop"]=list_top[i_]
        series.append(temp)
    return series
```

调用前面三个函数，得到返回的三个列表。遍历三个列表，将数据转化为[{name:省份名字,value:省份 ID,project985:985 数量,project211:211 数量,doubleTop:双一流数量},{…},…]的格式。

该函数是为地图可视化小组提供数据接口。但是因为使用 js 不方便调用，最终未被采用。

c.大学/专业录取分数和位次的图表展示

```
#输入学校，省份和科类，得到学校近三年的录取分数，录取位次和变化趋势
def getInfoOfUniv(college,pID,cID,major="all"):
    #参数为学校 ID，省份 ID（整数 1-34），科类 ID（整数 1-3）
    scoreList=[]
    rankList=[]
    for _i in range(2017,2020):
        result1=Majors.objects.filter(collegeID=college,
                                      provinceID=pID,
                                      categoryID=cID,
                                      year=_i,
                                      majorName__contains=major,)     #模糊查询
        if(result1.exists()):
```

```
            score=result1[0].minScore
        else:
            score=0
        scoreList.append(score)
        result2=Rankings.objects.filter(score=score,
                                        provinceID=pID,
                                        categoryID=cID,
                                        year=_i,)
        if(result2.exists()):
            rank=result2[0].rank
        else:
            rank=0
        rankList.append(rank)
    return scoreList,rankList
```

输入形参为学校 ID，省份 ID，科类 ID，专业名称（默认形参，默认值为 all，即大学录取线）。按年份遍历，使用 filter 方法查询出该年录取分数，再由分数查询出位次，查询结果存储到列表中。考虑到专业名称多变，所以采用包含关键词的模糊查询。判断 filter 返回的 querySet 是否为空，若为空，则取 0 值。将两个列表返回。

```
def drawPicture(List,name):
    years=[2017,2018,2019]
    plt.xlabel('year')
    plt.ylabel(name)
    plt.plot(years,List,linewidth=3, color='b', marker='o',
             markerfacecolor='blue', markersize=5)
    x_major_locator=MultipleLocator(1)     #设置 x 坐标轴的坐标间隔为 1
    ax=plt.gca()
    ax.xaxis.set_major_locator(x_major_locator)
    for _, score in zip(years, List):
        plt.text(_, score, score, ha='center', va='bottom', fontsize=10)
    #将生成的图表返回到前端
    sio = BytesIO()
    plt.savefig(sio, format='png')
```

```
    data = base64.encodebytes(sio.getvalue()).decode()
    src = '' + str(data)
    plt.close()
    return src
```

该函数接收数据和坐标轴名称，使用 matplotlib.pyplot 绘图。使用 BytesIO 和 base64 将图片解码，然后将解码后的数据返回。

```
def CollegeInfoGraphs(request):
    categoryDict={"文科":1,"理科":2,"综合":3}
    zeroList=[0,0,0]
    src1=drawPicture(zeroList,"score")
    src2=drawPicture(zeroList,"rank")
    if(request.method=="POST"):
        college = request.POST.get("college")
        category = request.POST.get("category")
        province=request.POST.get("province")
        provinceID=Provinces.objects.get(provinceName=province).provinceID
        collegeID=Colleges.objects.get(collegeName=college).collegeID
        scoreList,rankList=getInfoOfUniv(collegeID,provinceID,categoryDict[category])
        src1=drawPicture(scoreList,"score")
        src2=drawPicture(rankList,"rank")
        text1="图 1：近三年"+college+"录取分数线变化"
        text2="图 2：近三年"+college+"录取位次变化"
        return
render(request,"CollegeInfoGraphs.html",{"src1":src1,"src2":src2,"text1":text1,"text2":text2})
    else:
        return render(request,"CollegeInfoGraphs.html",{"src1":src1,"src2":src2})
```

该函数为前后端连接的函数，接收 request，并从前端界面接收用户输入的学校名称，科类名称，省份名称数据，查询得到其对应的 ID，根据 ID 调用 getInfoOfUniv()和 drawPicture()查询并生成图表返回到前端显示。实现专业分数线和位次查询功能的 MajorInfoGraphs()函数与之相似。

c.考生邻近省份的一流高校查询

#输入考生省份 ID，展示本省及相邻省份的双一流院校

```
def getTopUnivOfNeighbors(pID):
    #存储相邻省份的字典
    Regions={……}  #省略具体内容
    topDict={}
    Neighbors=Regions[pID]
    Neighbors.append(pID)
    for ID in Neighbors:
        pName=Provinces.objects.get(provinceID=ID).provinceName
        tops=Colleges.objects.filter(provinceID=ID,top=True)
        if(tops.exists()):
            topList=[]
            for top in tops:
                topList.append(top.collegeName)
            topDict[pName]=topList

    return topDict
```

该函数接收省份 ID，在 Regions 字典（键为省份 ID，值为邻近省份的 ID，邻近的标准为接壤）中取出其邻近省份 ID 的列表。遍历该列表，依次查询出这些省份一流高校的名称，存储到字典中。函数返回该字典（键为省份名称，值为存有该省份一流高校名称的列表）。

```
def DisplayTopUnivOfNeighbors(request):
    if(request.method=="POST"):
        province=request.POST.get("province")
        provinceID=Provinces.objects.get(provinceName=province).provinceID

        topDict=getTopUnivOfNeighbors(provinceID)

        return render(request,"TopUnivOfNeighbors.html",{"topDict":topDict})
    else:
        return render(request,"TopUnivOfNeighbors.html")
```

该函数接收 request，使用前端传回的用户输入的省份名称，调用 getTopUnivOfNeighbors()函数得到一流高校字典，传到前端显示。

d.页面设计

使用 UI 组提供的“祥龙咨询”模板。一共设计了三个页面 TopUnivOfNeighbors.html，MajorInfoGraphs.html，CollegeInfoGraphs.html，分别实现邻近省份一流高校查询，专业录取信息查询和学校录取信息查询。

使用形如<input type="text" placeholder="大学" name="college" required="" style="width: 200px">实现输入框接收用户输入。

按钮<button type="submit" value="OK">OK</button>

使用形如

```
<img src='{{src1}}'    width="400" height="300">
```

实现接收后端传来的解码图片并显示。

使用形如{% for province,list in topDict.items %}

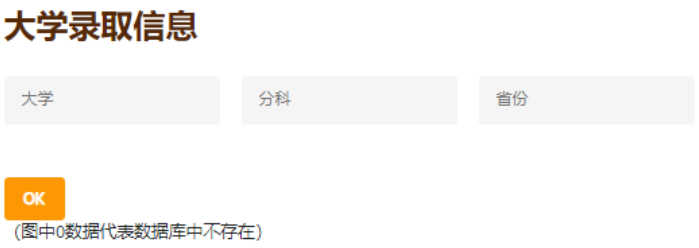
```
<h4 style="text-align:center">{{province}}:</h4>
{% for collegeName in list %}
    <p class="pos-abs" style="text-align:center">{{collegeName}}</p>
{% endfor %}
```

{% endfor %}

实现遍历字典并显示到前端上。

五、实验结果与分析

A．大学录取分数和位次的查询效果（输入复旦大学、理科、山东）



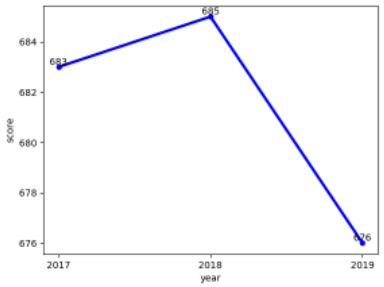


图1：近三年复旦大学录取分数线变化

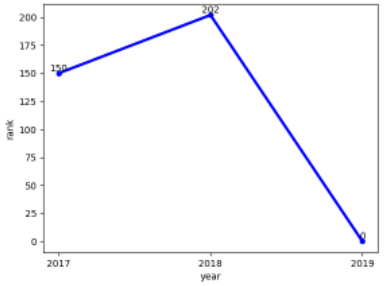


图2：近三年复旦大学录取位次变化

输入学校、科类、省份，可以得到近三年该学校的分数线和录取位次的变化，给予考生参考。

B. 专业录取分数和位次的查询效果（输入东华大学、计算机、理科、山东）

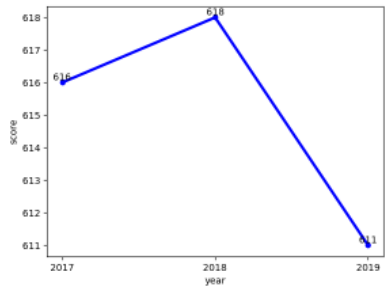


图1：近三年东华大学计算机专业录取分数线变化

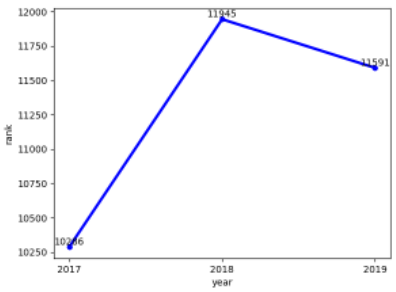


图2：近三年东华大学计算机专业录取位次变化

输入学校、专业、科类、省份，可以得到近三年该学校该专业的分数线和录取位次的变化，给予考生参考。

C. 邻近省份的一流高校查询结果（输入山东）

输入考生省份，可以查询出考生所在省份的邻近省份的所有一流大学。

六、实验总结与心得体会

本人在该次实验中深刻了解并体会到了基于 gitee 的团队开发模式，学到了 git 的相关理论知识。并且深刻体会了 Django 网站开发框架，使用 python 完成了数个网页从后端到前端的开发。在小组合作中，也与其它队员积极交流，带领数据库使用小分组解决了一些其它分组的数据需求。

在这次实践任务中，我也深深体会到了团队合作的不易：组内成员并不适应前后端分离的分工模式，导致很多小分组是在独立开发某一功能；而且很多队员在无意间挤占了其它分组的任务，将预定分工中其它组的任务自己独立完成。

另外，小组的任务比较模糊，在开发前期并没有明确认知和规划，需求分析不够具体可行，预定分工大家也没有严格执行，这些都导致我们预定的数据库使用小组任务量极少，最后只好自行思考创意并自己实现，以达到合适的工作量。

关于团队合作的总结，需求和分工都要明确可行，每个分组都要积极寻求合作解决问题，而不是各自为战；代码的正确性和各分组的任务完成度也需要组长和各小组长监督到位，出现合作不畅也要及时沟通，将问题消灭在萌芽之中。