暑期学校实验项目：高考志愿填报助手

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 小组名称 | Django开发 | | | | | | |
| 姓 名 | 朱斌 | 专业 | 人工智能 | 班级 | （2）班 | 学号 | 09118207 |
| 实验时间 | 2020.8.31-2020.10.7 | | 指导教师 | 孔祥龙 | | 成绩 |  |
| **一、实验背景和目的**  高考是人生的一个重要里程碑，而志愿填报更是需要对未来人生的规划。放假在家，邻居孩子跑来问你，我这成绩能考上东南大学吗？如果考不上东大，我能考上哪？作为计算机类专业学生，我们通过收集各高校的录取分数数据和各省市近三年文理科排名来设计一个基于Web的高考志愿推荐系统，以解决这一问题，并将系统作进一步精确和完善以便后人使用。 | | | | | | | |
| **二、小组任务和个人任务**  本小组主要是通过对接其他小组的算法和数据，通过Django框架下的ORM操作来搭建一个包含“大学”、“专业”、“省份”、“科类”、“一级学科”、“分数排名”六个实体对象的数据库，并将数据导入其中以供其他小组查询和使用，同时负责对接其他小组对数据的修改需求并对数据库作进一步修改。  我在小组中所完成的任务为创建登陆界面，代码搭建数据库，创建数据表格，导入数据，修改数据，并一次又一次地更新和完善数据库，最终提供一个数据比较精确和完善的db.sqlite3给其他小组。 | | | | | | | |
| **三、个人任务需求分析**  创建一个拥有六个实体的数据库，将第一组给出的数据整理后导入数据库对应实体中，并将数据库接口提供给各组以便查询和使用。同时，对接其他组成员并对数据库数据进行修改。  使用ORM操作，只需要面向对象编程, 不需要面向数据库编写代码。对数据库的操作都转化成对类属性和方法的操作。不用编写各种数据库的sql语句。实现数据模型与数据库的解耦, 屏蔽了不同数据库操作上的差异。根据实际情况任选数据库使用sqlite。  数据库包含六个实体对象，分别是“大学”、“专业”、“一级学科”、“省份”、“分数排名”、“科类”。大学类中包含六个属性，其中“大学ID”为主键，“省份ID”为省份类的外键。专业类中包含九个属性，其中“大学ID”为大学类的外键，“省份ID”为省份类的外键......以此类推。其中，专业类的属性“一级学科IDs”表示该专业对应的一级学科，可以没有也可以有多个，所以这里的一级学科是一个文本数据，多个ID由特定字符分开。因此，这里的一级学科IDs与一级学科类中的一级学科ID不同，也不是外键。 | | | | | | | |
| 1. **实验过程（需附上关键代码及相关说明）**   (1)熟悉Django的安装和使用，并实现基本交互功能。  (2)搭建数据库。(具体代码可见ormDesign/models.py)  class Colleges(models.Model): # 大学类  collegeID = models.IntegerField(primary\_key=True) # 大学id（主键）  collegeName = models.TextField() # 大学名称  provinceID = models.ForeignKey('Provinces',  on\_delete=models.DO\_NOTHING,  null=True) #省份id（外键->Provinces类）  project985 = models.BooleanField(default=False) #是否为985大学  project211 = models.BooleanField()#是否为211大学  top = models.BooleanField() #是否为一流学校  class Meta:#设置元信息，使用db\_table自定义表的名字  db\_table = 'Colleges'  verbose\_name = '大学列表'  注：这是“大学”实体的数据库，其中collegeID为主键，provinceID为外键，并且是“省份”实体的主键，还有一些其他的属性。并且通过python manage.py makemigrations在ormDesign内的migrations包内将类转换为数据表结构，再通过python manage.py migrate运行操作文件，创建数据表db.sqlite3。  (3)向数据库导入数据。(具体代码可见ormDesign/views.py以及ormDesign/urls.py)  注：以大学为例，在views.py里定义插入数据的getColleges函数  def getColleges(request):  df = pd.read\_csv('./extFiles/college.csv', encoding="gbk")  num = df.shape[0]  for x in range(num):  c2 = models.Colleges(collegeID=df.at[x, 'collegeID'],  collegeName=df.at[x, 'collegeName'],  provinceID=models.Provinces.objects.get(provinceName=df.at[x,'province']),  project985=df.at[x, '985'],  project211=df.at[x, '211'],  top=df.at[x, 'top'])  c2.save()  return HttpResponse("College in")  然后在urls.py里填写调用getColleges函数的路径，并最终通过网页交互操作实现函数调用，从而导入数据。  urlpatterns = [  path('1/2/3', views.getColleges, name='college'),  ]  最终网页出现“College in”即表示大学数据全部导入成功。  (4)与第一组一同修改和完善数据，并对数据库进行新一轮的删除和修改。  注：前期数据库更新是通过代码删除与更新，后期部分数据删除直接通过db.sqlite3数据表进行操作。 | | | | | | | |
| **五、实验结果与分析**  本小组最终的实验成果为一份15.8M内存的数据表db.sqlite3，其中包含我们所设计的六个实体的数据，在这之中，专业类拥有约18万条数据(后续在db.sqlite3中的直接修改删除部分非本一线数据未作计算)，分数排名类拥有约2万9千条数据。  我们提供的数据库可以通过代码访问读取数据，也可以通过Navicat软件便捷地查看数据库数据。里面的外键可以通过关联操作获得它想表达的含义。其他小组最终通过读取我们数据库的数据来实现他们知识图谱的构建或者相关算法的选取。 | | | | | | | |
| **六、实验总结与心得体会**  随着暑期学校最后一节软件实践课的结束，我们的项目暂时告一段落，但它还远远没有结束。这次的软件实践项目中，我们小组其实并没有去实现多么高端的算法，我们做的只是最基本的工作，创建数据库并将数据导入，同时对接其他小组的需求。但是，我们的工作也是非常重要的工作，因为除了第一小组，其他小组的工作都需要基于我们数据库的完成。我还记得第二周的时候，早上刚讲完数据库搭建的相关知识，下午回去后，我们小组就努力完成了数据库搭建的相关代码并将其pull到老师的仓库，以供其他组设计算法时作为参考。我们花了将近两天的时间将初始的数据导入数据库(因为通过网页操作导入数据真的非常慢)，总的来说大家都还是挺积极努力的，在组长的带领下后续对数据的陆陆续续修改也积极配合。  但是，也出现了不少问题。其中最麻烦的就是数据的缺失和错误。因为这是在上学期我们大家爬取时就出现了部分问题，后来第一组和我们组也没有细看，先行将其导入了数据库。问题开始发现是在第二周快要结束的时候，一个算法组跟我们沟通说他们算法算出来有问题，后来经过一系列的检查后，我们才发现是某个大学的数据出了问题，于是准备整改，然后一个又一个的问题又接着暴露了出来。后两周的时间我们大多都是在修改错误数据和补全数据。同时也很感谢孔老师对我们的指导和提出的部分建议，最终由于时间和精力有限，我们只能将一份比较完善的数据库上传到了老师的仓库。  这次的项目可能是我大学三年以来参加过的最大的一个项目，它拥有9个小组，100多名成员，同时它也让我获得了很多大项目构建的经验。比如说我们的数据库搭建，其实其它组可以从csv文件中直接读取数据，但是由我们的数据库更方便数据的管理和读取速度，同时也为他们提供了一个很好的反映数据问题的接口，这使得我们整个项目更加正规和系统。同时，在这次项目中，我也认识到了合作的重要性，不仅仅是小组成员之间的合作，小组与小组之间的合作也非常重要。同时，初始数据的可靠性也非常重要。我们不可能做到初始给出的数据就是完美的，但我们在开始项目前可以尽可能地保证初始数据的可靠性，这样无论是对于项目本身还是后续的修改完善，都是有好处的。最后，我很感谢这次的项目让我学到了很多，也让我感到有所成果，我希望它能够更好地延续和完善下去。 | | | | | | | |

2020年9月制