暑期学校实验项目：高考志愿填报助手

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 小组名称 | 知识图谱构建B组 | | | | | | |
| 姓 名 | 黄腾 | 专业 | 吴健雄学院 | 班级 | 615181 | 学号 | 61518125 |
| 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**  问题场景：放假在家，邻居孩子跑来问你，我这成绩能考上东南大学么？如果考不上东大，我能考上哪？  目的：建立高考志愿推荐的系统，可根据提供的分数及排名信息推荐最优的志愿填报，即根据全国各高校往年的最低录取分数线等信息利用相关算法得到推荐上的大学。 | | | | | | | |
| **二、小组任务和个人任务**  小组任务：构建关于省份的知识图谱，导入数据，建立关于省份信息的数据库，确定实体之间的逻辑关系，提取数据模式中的RDF三元组，设计表的模式，基于现有数据库进行存储，三元组提取模块及html数据可视化模块。  个人任务：和小组成员一起设计数据库模式，利用一组的提供的数据和neo4j图数据库构建关于省份的知识图谱。 | | | | | | | |
| **三、个人任务需求分析**  处理并利用一组提供的数据，读取csv, json等文件并处理。根据省份，大学，分数线等数据格式和相互之间的关系设计数据库模式，学习neo4j数据库的相关知识并利用neo4j数据库构建关于省份的知识图谱。为前端设计可视化页面提供后端接口。 | | | | | | | |
| **四、实验过程（需附上关键代码及相关说明）**    数据库设计：    基本配置：配置neo4j图数据库。下载py2neo包。  利用json文件和neo4j数据库的相关知识建立关于省份的知识图谱。  创建省份节点：  provinceNode = [] # 存储创建好的34个省份节点，用于后续创建关系 for i in range(provinceinfo.shape[0]):  provinceNode.append(Node('省份', ID=int(provinceinfo.iloc[i][0]), Name=str(provinceinfo.iloc[i][1])))  graph.create(provinceNode[i])  创建学校节点：  collegeNode = [] # 存储创建好的学校节点，用于后续创建关系 for i in range(collegeinfo.shape[0]):  collegeNode.append(Node('学校', ID=int(collegeinfo.iloc[i][6]),  Name=str(collegeinfo.iloc[i][2]), \_985\_=int(collegeinfo.iloc[i][3]),  \_211\_=int(collegeinfo.iloc[i][4]), top=int(collegeinfo.iloc[i][5]),  ProvinceID=int(collegeinfo.iloc[i][0])))  graph.create(collegeNode[i])  创建分数信息节点：  scoreNode = [] # 存储创建好的学校节点，用于后续创建关系 for fileName in scorefiles:  if not os.path.isdir(fileName): # 判断是否是文件夹，不是文件夹才打开  scoreinfo = pd.read\_csv('json\_csv' + '/' + fileName, encoding='utf-8') # 加载csv文件数据   # 属性:值  provinceID = int(scoreinfo.iloc[0][3])  year = int(scoreinfo.iloc[0][4])  category = str(scoreinfo.iloc[0][5])  scoreLineClass = [] # 分数线类别:一本线、二本线、专科线...  scoreLineValue = [] # 分数线取值(与上面的列表对应)  score = [] # 一分一段表的分数  cumulateNumber = [] # score对应的累计人数   # 省份分数线（一本线、二本线、专科线...）  scoreLineClass = list(scoreLine[provinceinfo.iloc[provinceID-1][1]][str(year)][category].keys())  scoreLineValue = list(scoreLine[provinceinfo.iloc[provinceID-1][1]][str(year)][category].values())   # 分数list、累计人数list  for i in range(scoreinfo.shape[0]):  score.append(int(scoreinfo.iloc[i][1]))  cumulateNumber.append((int(scoreinfo.iloc[i][2])))   scoreNode.append(Node('分数信息', provinceID=provinceID, year=year,  category=category, score=score, cumulateNumber=cumulateNumber,  scoreLineClass=scoreLineClass, scoreLineValue=scoreLineValue))  graph.create(scoreNode[-1])  创建学校和节点之间的关系：  for i in range(provinceinfo.shape[0]):  for j in range(collegeinfo.shape[0]):  if(collegeinfo.iloc[j][0] == provinceinfo.iloc[i][0]): # 学校属性.省份ID == 省份属性.ID  graph.create(Relationship(provinceNode[i], 'have', collegeNode[j]))  graph.create(Relationship(collegeNode[j], 'located', provinceNode[i]))  创建分数信息和省份之间的关系：  for i in range(len(scoreNode)):  for j in range(provinceinfo.shape[0]):  if(scoreNode[i]['provinceID'] == provinceinfo.iloc[j][0]):  graph.create(Relationship(scoreNode[i], 'score\_province', provinceNode[j]))  graph.create(Relationship(provinceNode[j], 'province\_score', scoreNode[i])) | | | | | | | |
| **五、实验结果与分析**  省份节点：    学校节点：    学校和省份之间的关系：      可以看到，省份与学校节点已经建立，学校与对应省份之间的关系也已经建立起来 | | | | | | | | |
| **六、实验总结与心得体会**  本次实验是一个一百多人共同参与的大项目，本组负责的只是其中的一部分，具体到个人就更只是一小部分的工作。更多地体会到的是多人合作开发的方法和主要流程，利用码云对代码的管理流程。同时学到了关于web开发，djingo框架，neo4j数据库等相关知识。今后应对多人合作进行的工程项目等有更高层次的认识。 | | | | | | | | |

2020年9月制