《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	知识图谱构建 A 组						
姓　　名	蒋林煊	专业	人工智能	班级	091181	学号	09118142
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：

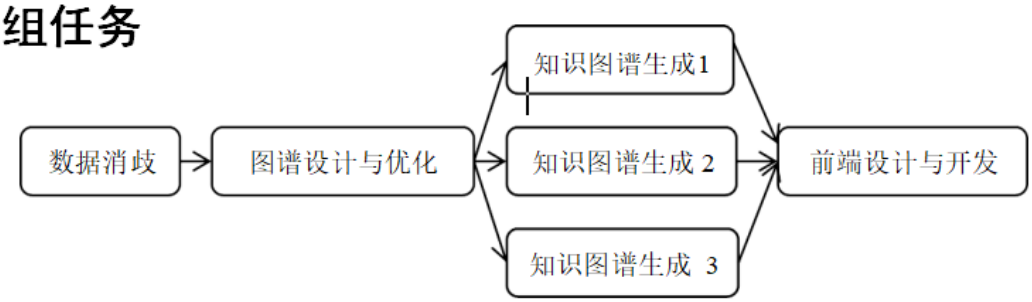
高考是人生的重要关口，如何选择自己最适合的学校是历年各个考生都关注的重要问题。如何选择专业，如何观察历年录取分数线变化，如何了解各个学校在不同省份的录取分数差异，是每年高考结束以后学生与家长共同的关注焦点。

实验目的：

通过构建知识图谱，训练人工智能算法实现出一个推荐算法，满足学生与家长对高考志愿填报的咨询需求

二、小组任务和个人任务

小组任务



任务 1：数据消歧

本项目需要用到的数据源，是第一组清洗的包含学校，专业，省份，分数，年份的 csv 文件。由于专业名称等信息存在相同专业不同名称等现象，需要先进行消歧工作。

任务 2：知识图谱设计与优化

利用已有的数据构建一个小型的报考知识图谱(知识库)，通过调用该图谱可以实现如下功能：

1. 已知自己某分数能上什么学校
2. 某个特定的专业哪个学校分数最高
3. 已知自己的分数判断自己能学什么样的专业
4. 查询某学校的特定专业
5. 我只想学 XX 专业，能去什么学校？

任务 3：知识图谱生成

1. 对所有实体生成可以导入Neo4j 的 csv 文件
2. 对所有关系生成可以导入Neo4j 的 csv 文件
3. 将以上文件导入 neo4j，生成知识图谱
4. 再根据图谱，改进不足

任务 4：基于构建好的知识图谱，构建显示网页

此部分属于前端操作，主要考虑图谱可视化效果

个人任务(组内我和张骥同学又分为同一任务组)

任务 1：创建可以导入Neo4j 的部分 csv 文件

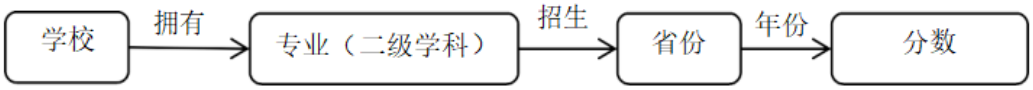
任务 2：将可以导入的 csv 文件导入 neo4j,初步形成知识图谱

三、个人任务需求分析

任务 1：创建可以导入Neo4j 的部分 csv 文件

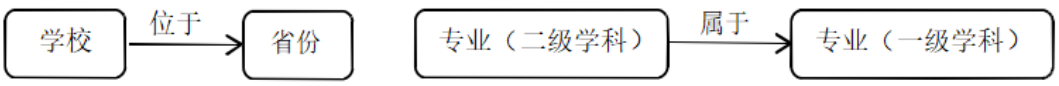
其他同学已完成知识图谱设计，并且对原始数据进行消歧以及命名主键等工作， 在本任务中，我的任务是从预处理后的数据，提取出实体、关系，创建出可以导入的 csv 文件。

在知识图谱中，每个节点就是实体，节点之间的联系就是关系。由于我们组是从 学校入手，所以我们设计了以下图谱：



其含义为：某学校拥有多个专业，该专业在不同省份招生，不同省某年该专业不同学校的分数线。我的具体工作是，创建分数实体，包含文理科、学校、专业等属性； 创建省份-分数关系，关系属性包含年份。

此外，还可以添加以下关系：



这两个关系已存在，只需导入即可。

四、实验过程（需附上关键代码及相关说明）

```
import os
import csv
import hashlib

#生成 md5 码用来作为主键（唯一标识符）

def get_md5(string):
    """Get md5 according to the string
    """
    byte_string = string.encode("utf-8")
    md5 = hashlib.md5()
    md5.update(byte_string)
    result = md5.hexdigest()
    return result


#将 college 的 csv 文件转化为可以导入 neo4j 形式的文件，格式见下一行

def build_college(college_prep, college_import):
    """Create an 'college' file in csv format that can be imported into Neo4j.
    format -> college_id:id name fame label
    label->college
    """
    print('Writing to {} file...'.format(executive_import.split('/')[-1]))
    with open(college_prep, 'r', encoding='utf-8') as file_prep, \
            open(college_import, 'w', encoding='utf-8') as file_import:
        file_prep_csv = csv.reader(file_prep, delimiter=',')
        file_import_csv = csv.writer(file_import, delimiter=',')
        headers = ['college_id:ID', 'name', 'fame', ':LABEL']
        file_import_csv.writerow(headers)
```

```
        for i, row in enumerate(file_prep_csv):
            if i == 0 or len(row) < 3:
                continue
            info = [row[0], row[1], row[2]]
            # info_id = get_md5('{},{},{}'.format(row[0], row[1], row[2]))    不用 md5 了
            info.insert(0, info_id)
            info.append('College')
            file_import_csv.writerow(info)
    print('- done.')

#将 major 的 csv 文件转化为可以导入 neo4j 形式的文件，格式见下一行

def build_major(college_prep,major_prep, major_import):
    """Create an 'major' file in csv format that can be imported into Neo4j.
    format -> major_id:ID,Year,Province,category,Major,Score,Contributer
    label -> major
    """
    print('Writing to {} file...'.format(stock_import.split('/')[-1]))
    major = set()
    with open(college_prep, 'r', encoding='utf-8') as file_prep:
        file_prep_csv = csv.reader(file_prep, delimiter=',')
        for i, row in enumerate(file_prep_csv):
            if i == 0:
                continue
            name = '{},{}'.format(row[0], row[1].replace(' ', ''))
            major.add(name)
    with open(major_prep, 'r', encoding='utf-8') as file_prep:
        file_prep_csv = csv.reader(file_prep, delimiter=',')
        for i, row in enumerate(file_prep_csv):
            if i == 0:
                continue
```

```
                _name = '{},{}'.format(row[0], row[1].replace(' ', ''))
                major.add(_name)
    with open(major_import, 'w', encoding='utf-8') as file_import:
        file_import_csv = csv.writer(file_import, delimiter=',')
        headers = ['major_id:ID', 'Year', 'Province', 'Contributer',':LABEL']
        file_import_csv.writerow(headers)
        for m in major:
            split = m.split(',')
            ST = False    # ST flag
            states = ['*ST', 'ST', 'S*ST', 'SST']
            info = []
            for state in states:
                if split[1].startswith(state):
                    ST = True
                    split[1] = split[1].replace(state, '')
                    info = [split[0], split[1], split[0], 'major']
                    break
                else:
                    info = [split[0], split[1], split[0], 'major']
            file_import_csv.writerow(info)
    print('- done.')

#将 college 和 prov 的 csv 文件转化为可以导入 neo4j 形式的文件，格式见下一行

def build_college_province(college_prep, relation_import):
    """Create an 'college_province' file in csv format that can be imported into Neo4j.
    format -> :START_ID,title,:END_ID,:TYPE
                college                province
    type -> located_in
    """
    with open(college_prep, 'r', encoding='utf-8') as file_prep, \
```

```
            open(relationlation_import, 'w', encoding='utf-8') as file_import:
        file_prep_csv = csv.reader(file_prep, delimiter=',')
        file_import_csv = csv.writer(file_import, delimiter=',')
        headers = [':START_ID', 'ID', ':END_ID', ':TYPE']
        file_import_csv.writerow(headers)
        for i, row in enumerate(file_prep_csv):
            if i == 0:
                continue
            # generate md5 according to 'name' 'gender' and 'age'
            start_id = get_md5('{},{},{}'.format(row[0], row[1], row[2]))
            end_id = row[3]    # code
            relation = [start_id, row[4], end_id, 'located_in']
            file_import_csv.writerow(relation)

#将 major 和 prov 的 csv 文件转化为可以导入 neo4j 形式的文件，格式见下一行

def build_prov_major(prov_prep, major_prep, relation_import):
    """Create an 'stock_industry' file in csv format that can be imported into Neo4j.
    format -> :START_ID,:END_ID,:TYPE
                major      prov
    type -> has
    """
    with open(prov_prep, 'r', encoding='utf-8') as file_prep_1, \
            open(major_prep, 'r', encoding='utf-8') as file_prep_2, \
            open(relation_import, 'w', encoding='utf-8') as file_import:
        file_prep_1_csv = csv.reader(file_prep_1, delimiter=',')
        file_prep_2_csv = csv.reader(file_prep_2, delimiter=',')
        file_import_csv = csv.writer(file_import, delimiter=',')
        headers = [':START_ID', ':END_ID', ':TYPE']
        file_import_csv.writerow(headers)
```

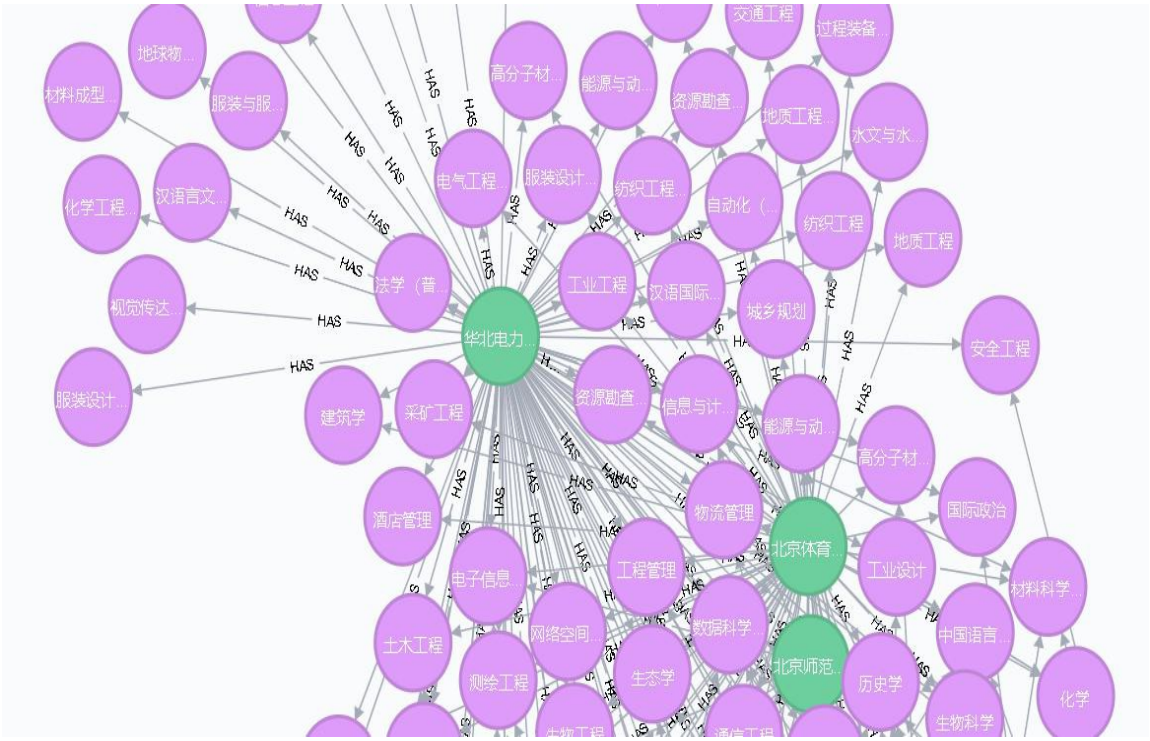
```
for i, row in enumerate(file_prep_1_csv):
    if i == 0:
        continue
    concept = row[1]
    start_id = row[1]
    end_id = get_md5(concept)
    relation = [start_id, end_id, 'has']
    file_import_csv.writerow(relation)


for i, row in enumerate(file_prep_2_csv):
    if i == 0:
        continue
    concept = row[2]
    start_id = row[0]
    end_id = get_md5(concept)
    relation = [start_id, end_id, 'has']
    file_import_csv.writerow(relation)
```

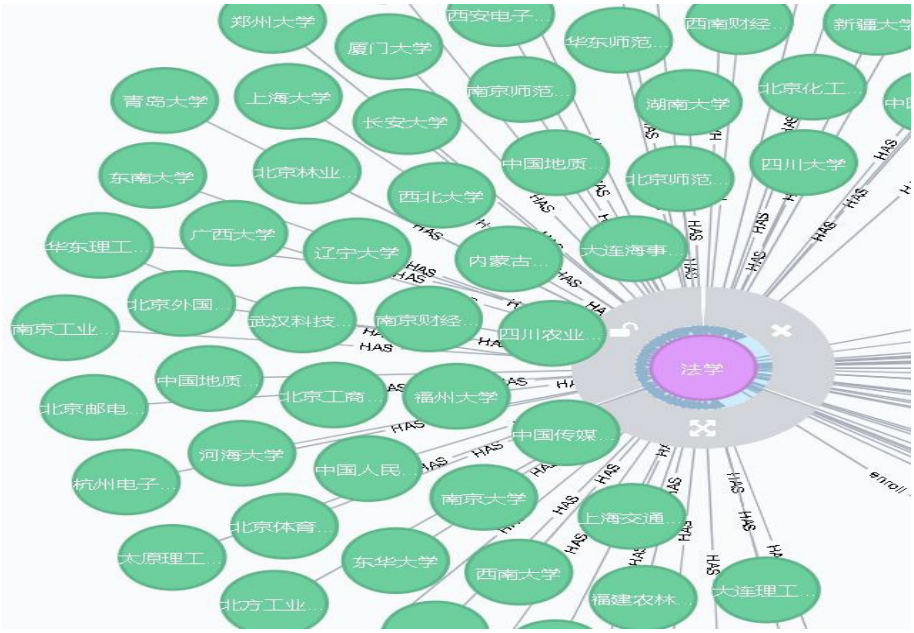
五、实验结果与分析

初始可以打开学校视图，如图，显示不同的学校

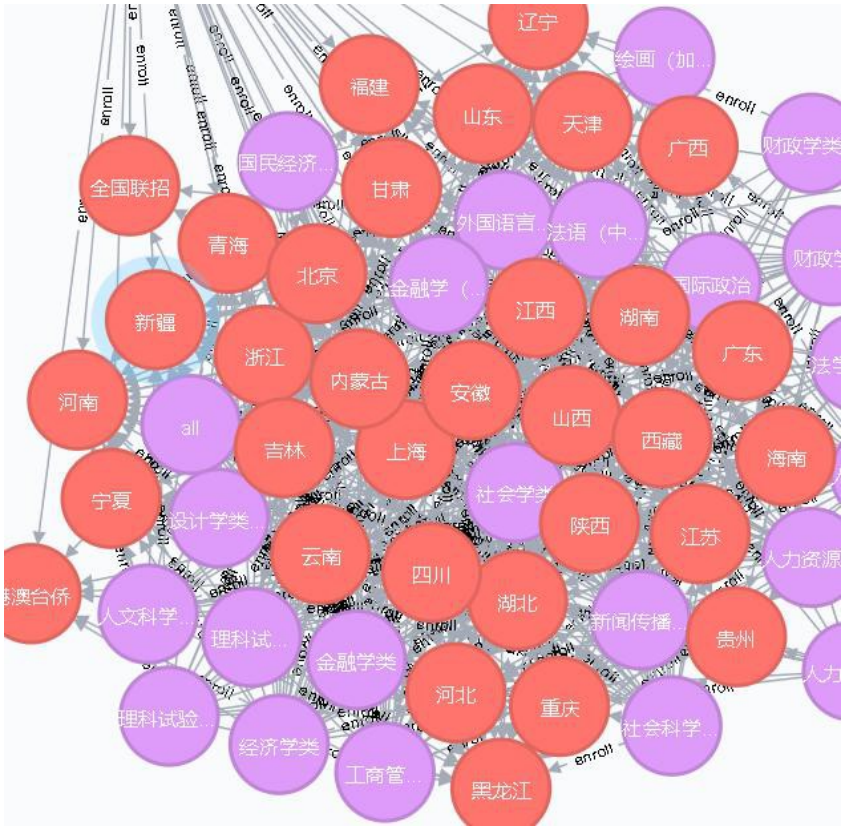
点击一个学校，可以看到它拥有的专业（HAS）



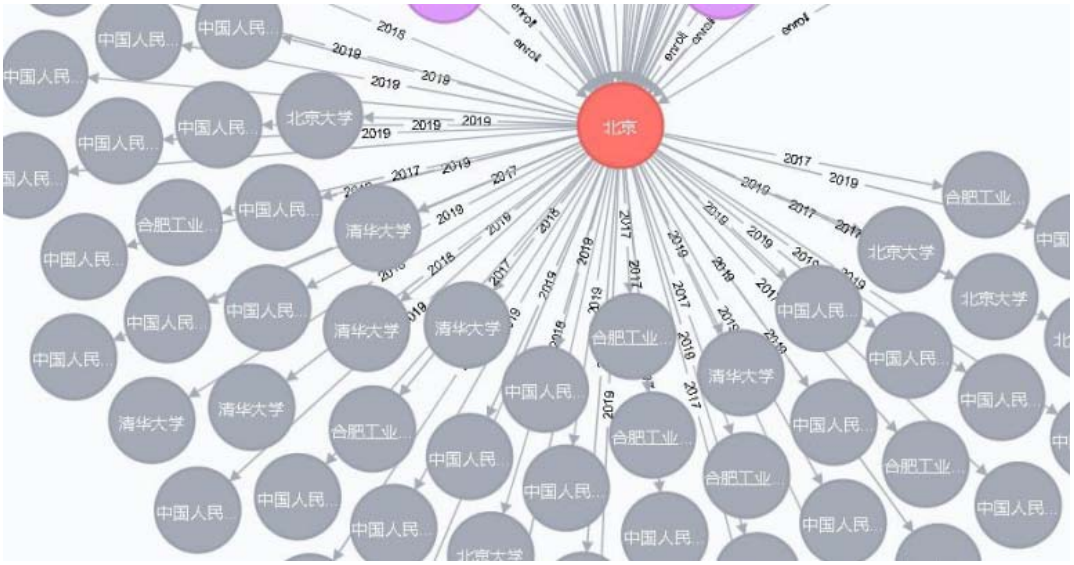
点击某一专业，如法学，可以看到拥有该专业的学校



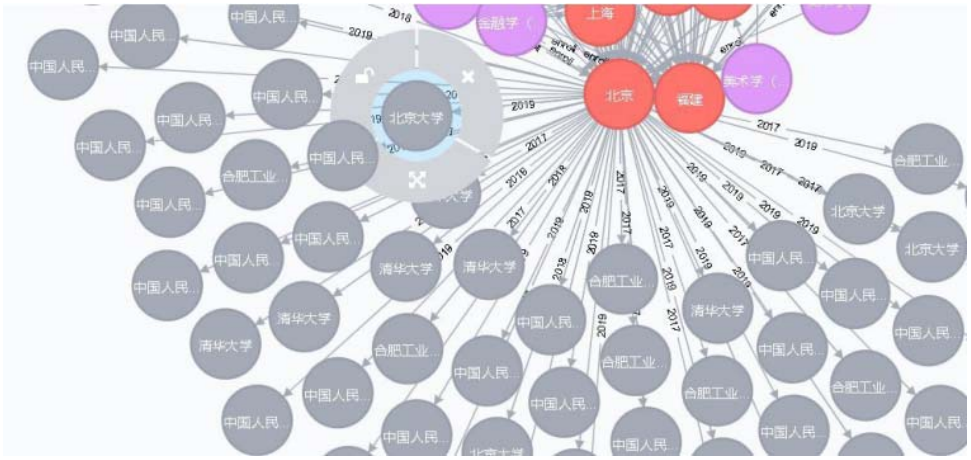
也可以看到它在哪些省份招生



点击北京省，可以看到北京省该专业某一年对应分数线



鼠标移动至某一节点，可以在属性栏看到专业、学校、文理科、分数等信息



分析：

由于我们有四类节点（学校、专业、省份、分数），所以可以有四个视图。初始视图都是一个一个独立的节点，在选择一个节点后，会出现许多与之关联的节点。一次选择，显示还是比较清楚，但是在两次以上的选择后，会显示大量节点与关系，不仅出现卡顿，而且会有大量数据聚集导致可视化效果很差。一个现有的方法是在选择节点后切换视图，清空之前

的显示。

此外，我们所希望的是点击学校后，显示该学校专业，点击专业，显示该学校该专业的招生省份，即对节点的选择存在向前的包含关系。经过观察，我们发现，在点击一个节点后，再点击一个节点，所有与之相连的节点都会显示，即我们选择学校的专业，显示的还有其他拥有该专业的学校，而该专业连接的省份， 是所有学校该专业招生的省份，省份与分数的关系连接与此类似。这与我们的预期相反。

六、实验总结与心得体会

这次实验，我和张骥的任务分别是处理数据和导入 neo4j，我们照着 GitHub 上一个完整的知识图谱项目学习，模仿处理数据，设计了新的 ER 图，体会到不同的设计模式对最终的结果有着很大的影响，所以在今后的开发过程中我们一定会在最开始设计好顶层的规划，这对我们之后的工作至关重要。通过这次实验，我们也体验了一次多人项目实现的流程，学会了多人合作下 git 仓库的使用，同时也切身体会到不同团队之间，同一团队的成员之间即时沟通和协商合作的重要性。这次课程的经验将为我以后的项目经历提供很大的帮助。

整个实验整体人数众多，故分配任务自然尤为重要。组与组之

间相互联系，上一组的工作稍有瑕疵，便另我们组的进展缓慢，进而进行组与组之间的沟通交流，由于第一组的数据清洗工作进度较缓，导致我们组的进度也较缓。这既让我对团队工作有了更深刻的认识，也理解了沟通的重要性。在之后我和张骥同学的团队协作中，因为也是室友，所以我们可以随时方便地进行沟通，时刻有着统一的目标，有了困难也可以一起解决。让我明白了协助的重要性。下次如果和其他同学分到一个小任务，我也学会了多利用交流软件如腾讯会议，zoom 等，及时的沟通才能少走弯路。

在转化 csv 文件中，一开始我们直接看官方 neo4j 的文档是一头雾水，所以我们在 GitHub 上找到类似的知识图谱项目，学着他们的转化代码，使我们受益匪浅，大大提高了效率，让我们明白了不要闭门造车，多多学习别人的长处，这对于我们也是很有利的。

最后经过我们百余人的努力，构建了这一个推荐系统，希望可以为以后的考生多做些贡献!