《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	唐云龙	专业	人工智能	班级	091181	学号	09118131
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

背景：每年都有大量高考毕业生需要进行志愿填报，对于他们而言，选择一个心仪的理想的学校的重要性是不言而喻的。

目的：我们项目的目的是帮助他们更好的了解自己的分数能够填报哪些学校，并根据他们的喜欢与特长推荐院校。同时通过这次实验，我们也能更好的提升自己需求分析、软件开发和持续集成的能力。在与组友合作的过程中也能更好的提升团队协作能力。

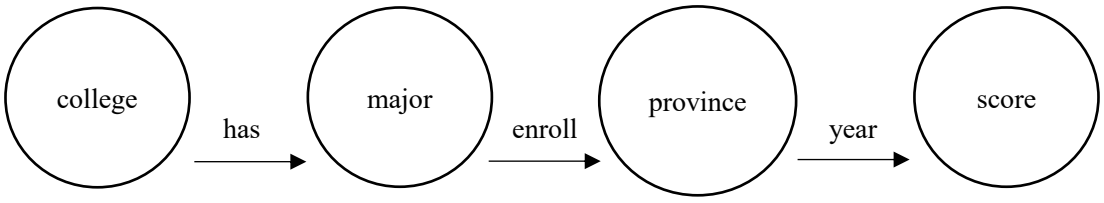
二、小组任务和个人任务

小组任务：1、获取数据源 2、知识图谱设计与优化 3、知识图谱数据准备 4、创建可以导入Neo4j 的 csv 文件 5、利用上面的 csv 文件生成数据库 6、基于构建好的知识图谱，构建显示网页

个人任务：参与设计知识图谱，创建可以导入Neo4j 的 csv 文件。

三、个人任务需求分析

在探讨知识图谱的设计过程中，我们设计出了如下图所示的知识图谱：



其中实体 college 中属性包括：学校 ID、学校名称、是否 985、标签为 College

实体 major 中属性包括：专业 ID、专业名称、标签为 Major

实体 province 中属性包括：省份 ID、省份名称、标签为 Province

实体 score 中属性包括：分数 ID、文理类别、分数、贡献者、学校、专业名称

关系 has 中包括：起始 ID 即学校 ID、终止 ID 即专业 ID、关系类别为 HAS

关系 enroll 中包括：起始 ID 即专业 ID、终止 ID 即省份 ID、关系类别为 enroll

关系 year 中包括：起始 ID 即省份 ID、终止 ID 即分数 ID、关系类别为年份

我们需要将创建的 csv 导入到 Neo4j 中，数据默认存放在 graph.db 文件夹里。服务重启之后，就可以通过 localhost:7474 观察到知识图谱了。

所以我需要创建出符合 Neo4j 输入的 csv。列名表示节点或关系的属性名，具体的列名有具体的 markup。对于实体，需要有该实体的:ID 用作查找连接后的节点和:LABEL 表示节点的标签。对于关系，需要有起始 ID 即:START_ID 和终止 ID 即:END_ID 表示关系文件中指向节点的 ID 列，还需要:TYPE 列说明关系的类型。还需添加其他相应属性列。

四、实验过程（需附上关键代码及相关说明）

在对其他组友做出的知识图谱进行研究后，我们提出了新的想法并付诸行动。在我们设计的图谱中，我们重新设计了数据的表达。从学校节点出发，一个学校会拥有许多专业。这里我们选取了二级专业。同时一个专业在不同地区会有不同的分数线，在同一个的地区的不同年份也会有不同的分数线。于是我们将省份与专业相连，代表着该专业在某些省份招生。而省份与分数相连，该关系中添加了年份属性，表示了这个分数线是该地区某年的分数线。

在 csv 的创建过程中，我主要对原来的 major 实体进行了修改，major 的修改牵连到了 has 关系与 enroll 关系的对应连接 ID 的改变，于是这两个的关系也被进行了修改。

```
def build_major(input_file, output_file):
    header_list = ['Major_id','Major',':LABEL']
    major_df = pd.read_csv(input_file,encoding = 'gbk')

    print('Writing to {} file...'.format(output_file.split('/')[-1]))
#       print(major_df[:11])
    major_id = major_df['Major:ID']
    print(type(major_id))
```

```
        major = major_df['Major'].drop_duplicates()  #对专业进行去重
        major = major.reset_index(drop=True)
        print(major.index)
#       major_imort = pd.concat([major_id,major],axis = 1)
#       major = major.to_frame()

        lst = ['M{}'.format(x) for x in major.index]
        lst = pd.DataFrame(lst).rename(columns={0:'Major:ID'})
#       lst = lst
#       print(lst[:10])
        major = pd.concat([lst,major],axis = 1)
        major[':LABEL']='Major'
        #major.to_csv(output_file, encoding = 'utf_8_sig',index=None)
        print(major[:10])
        print('- done.')
        return major
```

在本函数中，我对原来的 major 进行了提取。通过 header_list 可以看到，在 Major 这个实体中我们只需要 Major 的名称。将某专业在文理类都招生的情况放到分数 score 中体现，将省份以一个实体与专业相连，消去了专业实体中一个具体专业对应多条记录（一个专业在多地区招生）的情况，使知识表达更加简洁。所以此时的专业实体只表示该专业名称。因此我们可以大大减少 major 实体中记录的个数，从 16 万多条降至 2 千多条。在对专业进行去重后，每条记录对应一个单独的不重复的专业，在本函数中进行了重新编号。

下一步需要进行的是将新的 ID 与旧的 ID 对应起来。新 ID 以 M 开头，旧 ID 以 m 开头。我们可以知道，由于专业的去重，一个新 ID 对应着多个旧 ID。我们导入原来的 Major 表，将它与

新产生的 Major 表进行左连接，需要注意的是两表的 ID 列属性名需要不同，以两表的专业名称作为相同属性进行连接。

```
input_file=r'C:\Users\PC\Desktop\软件实践\二\major.csv'

major_ori = pd.read_csv(input_file,encoding = 'gbk')

df = major_ori.merge(major,how = 'left')
```

ID 匹配后的结果：

```
        Major:ID  Year Province category   Major  score  Contributor  \
0             m0  2019       p9       s2     all    615   09118101高捷
1             m1  2019      p29       s2     all    660   09118101高捷
2             m2  2019      p29       s1     all    629   09118101高捷
3             m3  2019       p1       s2     all    673   09118101高捷
4             m4  2019       p1       s1     all    637   09118101高捷
...          ...   ...      ...      ...     ...    ...           ...
162366   m163516  2017      p27       s1      英语    586  61518431郁航远
162367   m163517  2017      p27       s1  新闻传播学类    588  61518431郁航远
162368   m163518  2017      p27       s1   政治学类    585  61518431郁航远
162369   m163519  2017      p27       s1      哲学    585  61518431郁航远
162370   m163520  2017      p27       s1      法学    591  61518431郁航远

        disambiguated     ID    new :LABEL
0                 all   1304     M0  Major
1                 all   1304     M0  Major
2                 all   1304     M0  Major
3                 all   1304     M0  Major
4                 all   1304     M0  Major
...               ...    ...    ...    ...
162366       外国语言文学    502    M38  Major
162367        新闻传播学    503    M13  Major
162368          政治学    302  M1024  Major
162369           哲学    101   M535  Major
162370           法学    301     M3  Major

[162371 rows x 11 columns]
```

并删去多余属性后的结果：

	Major:ID	new
0	m0	M0
1	m1	M0
2	m2	M0
3	m3	M0
4	m4	M0
...	...	...
162366	m163516	M38
162367	m163517	M13
162368	m163518	M1024
162369	m163519	M535
162370	m163520	M3

[162371 rows x 2 columns]

由于 Major 表的 ID 更改，需要把关系 has 的:END_ID 和关系 enroll 的:START_ID 对应的更改。

关系 has 指的是学校拥有哪些专业。将匹配结果稍作修改，把 Major:ID 更名为:END_ID，为的是将原 has 表与匹配结果以相同的属性:END_ID 做连接。

	:START_ID	has	:END_ID	:TYPE	new
0	c10003	has	m1	HAS	M0
1	c10003	has	m2	HAS	M0
2	c10003	has	m3	HAS	M0
3	c10003	has	m4	HAS	M0
4	c10003	has	m5	HAS	M0
...	...	...	...	...	...
148123	c10730	has	m149221	HAS	M39
148124	c10730	has	m149223	HAS	M3
148125	c10730	has	m149224	HAS	M114
148126	c10730	has	m149225	HAS	M38
148127	c10730	has	m149226	HAS	M13

[148128 rows x 5 columns]

将:END_ID 列消去并将 new 属性更名为:END_ID，此时还需要对该表进行去重。操作后 has 表也变得简洁了很多。

	:START_ID	has	:TYPE	:END_ID
0	c10003	has	HAS	M0
1	c10001	has	HAS	M0
2	c10001	has	HAS	M1
3	c10002	has	HAS	M1
4	c10002	has	HAS	M2
...	...	...	...	...
7958	c10730	has	HAS	M1309
7959	c10730	has	HAS	M2241
7960	c10730	has	HAS	M2242
7961	c10730	has	HAS	M2243
7962	c10730	has	HAS	M612

[7963 rows x 4 columns]

类似的，实体 Province 也像 Major 一样，只保留省份名称信息。此时 enroll 表也易于获得，只需将上面的 ID 匹配后的表删去多余属性，留下 Province 和 new（即新的 ID）两个属性，Province 属性作为关系的:END_ID，new 作为关系的:START_ID，就可以得到 enroll 关系表。

	Province	new
0	p9	M0
1	p29	M0
3	p1	M0
5	p6	M0
7	p30	M0
...	...	...
162329	p27	M1478
162334	p27	M2544
162347	p27	M2533
162354	p27	M2531
162361	p27	M2536

[26964 rows x 2 columns]

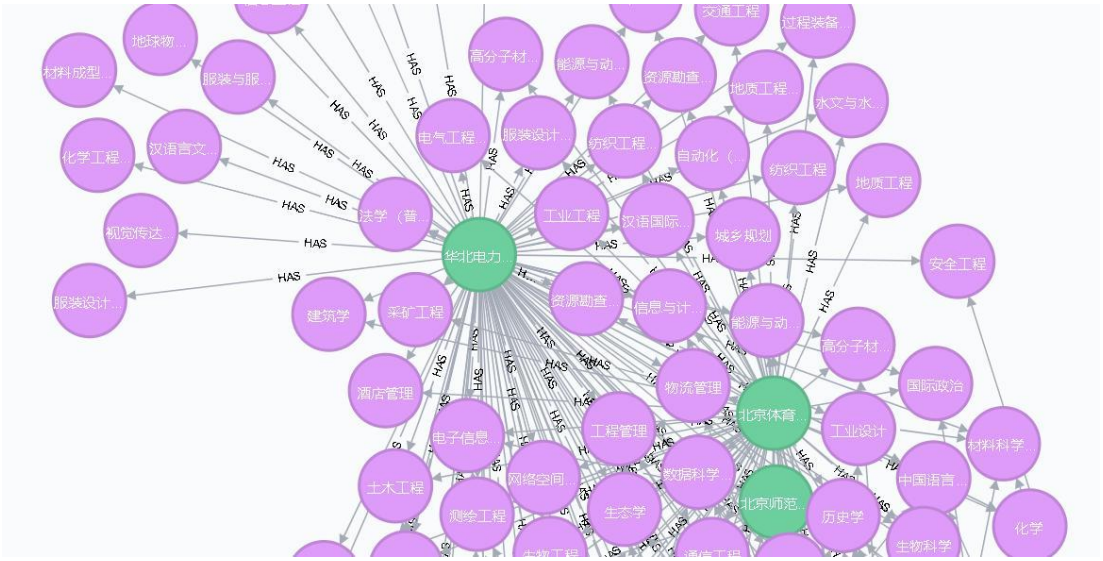
将创建好的 csv 使用 import 导入到 neo4j 中，即可看到图谱的效果。

五、实验结果与分析

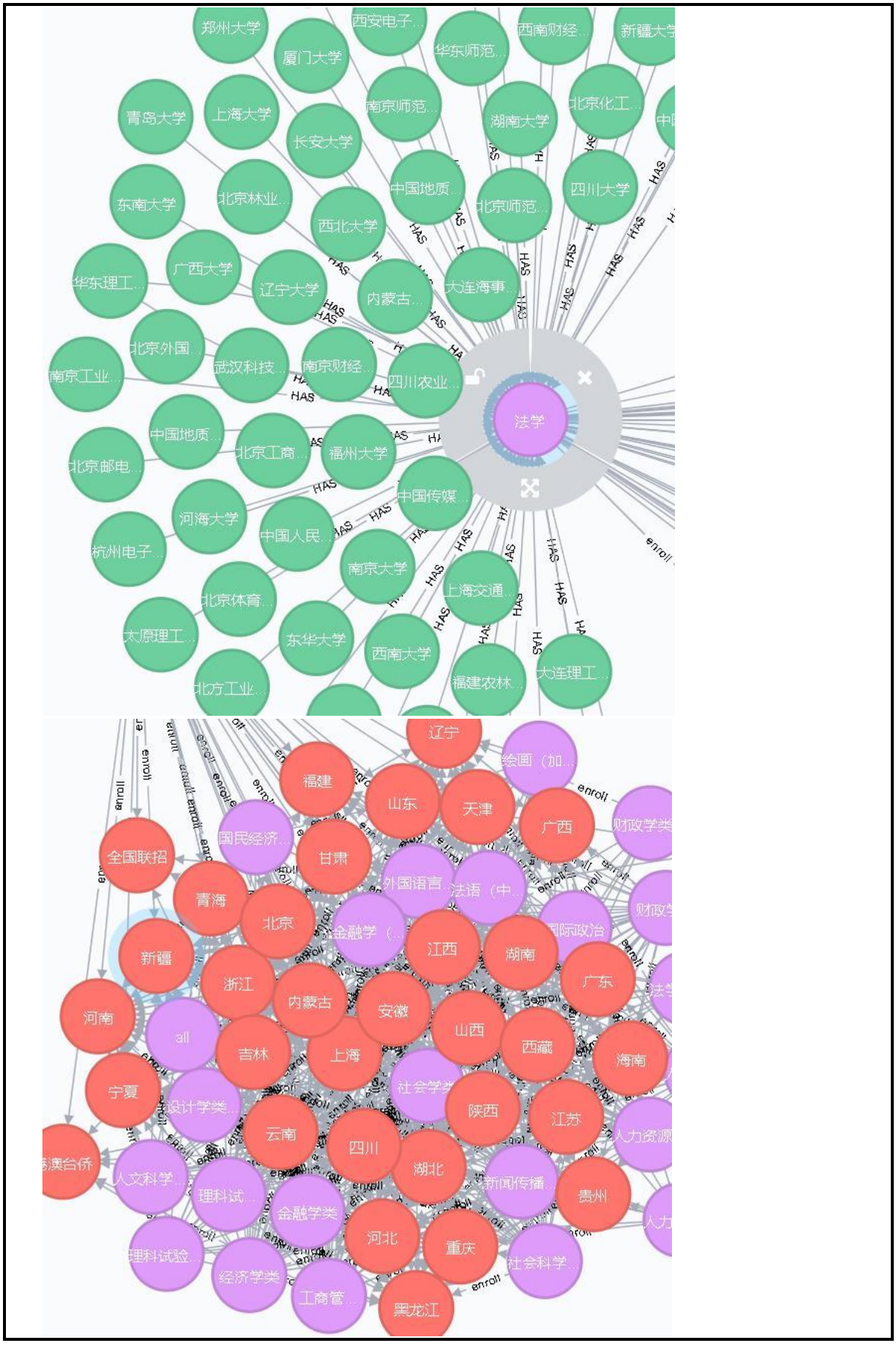
实验结果：

初始视图

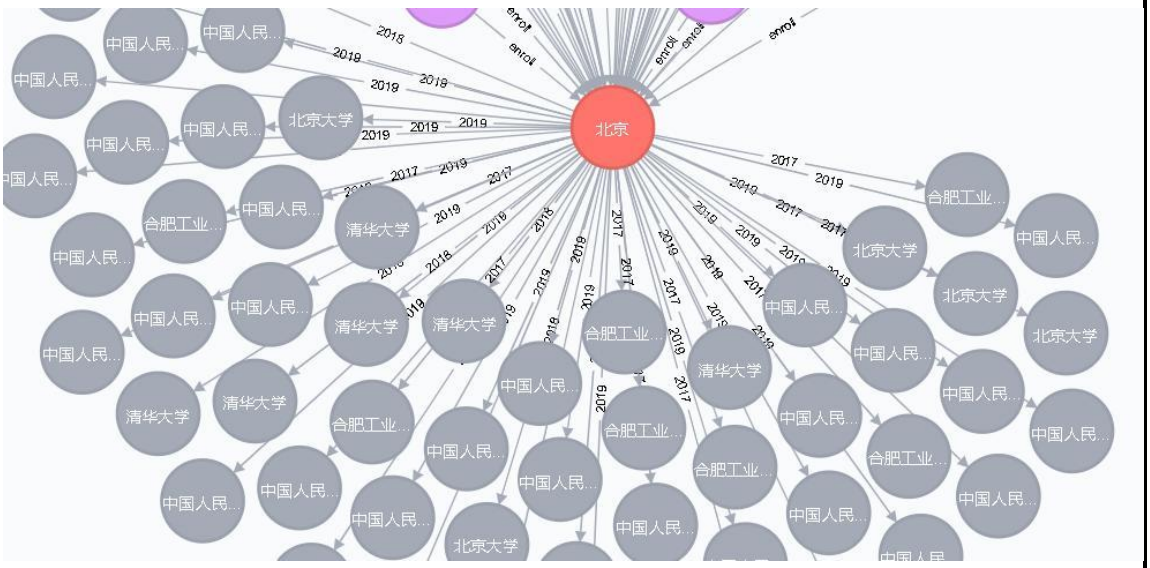
点开学校查看专业



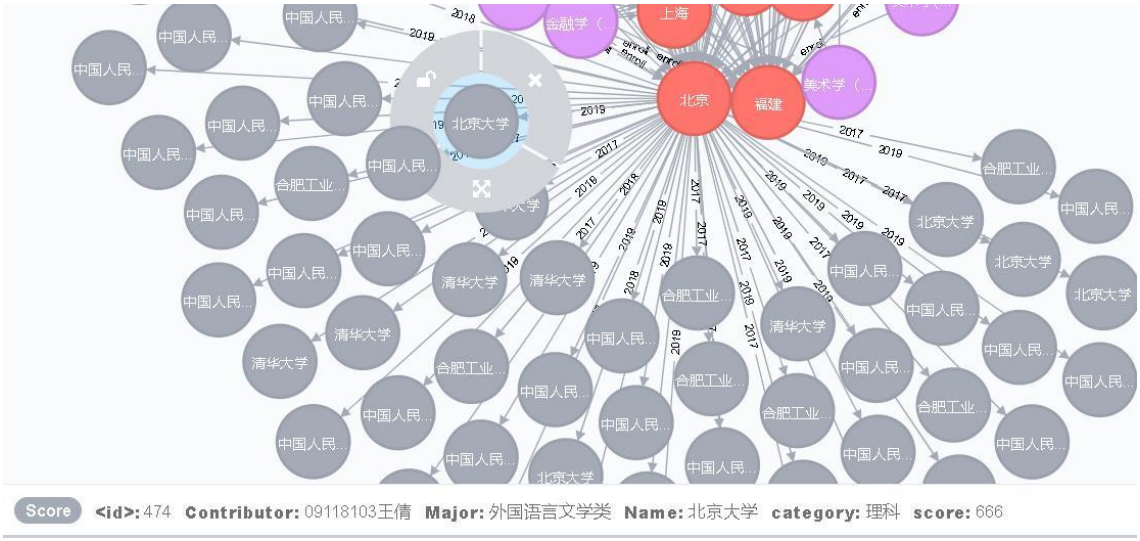
点击某专业则可以看到哪些学校有这个专业，也可以看到在哪些地区招生


郑州大学
厦门大学
西安电子...
华东师范...
西南财经...
新疆大学
青岛大学
上海大学
长安大学
南京师范...
湖南大学
北京化工...
东南大学
北京林业...
西北大学
中国地质...
北京师范...
四川大学
华东理工...
广西大学
辽宁大学
内蒙古...
大连海事...
北京外国...
南京工业...
武汉科技...
南京财经...
四川农业...
法学
中国地质...
北京工商...
福州大学
北京邮电...
中国传媒...
河海大学
中国人民...
杭州电子...
南京大学
上海交通...
北京体育...
太原理工...
东华大学
西南大学
大连理工...
北方工业...
福建农林...
HAS
enroll
辽宁
绘画（加...
福建
山东
天津
广西
财政学类
国民经济...
全国联招
甘肃
外国语言...
法语（中...
青海
北京
金融学（...
江西
湖南
新疆
浙江
广东
河南
内蒙古
安徽
山西
all
吉林
上海
西藏
海南
宁夏
社会学类
设计学类...
陕西
江苏
云南
四川
人力资源
港澳台侨
人文科学...
湖北
新闻传播...
理科试...
贵州
金融学类
河北
重庆
理科试验...
社会科学...
经济学类
工商管...
黑龙江

点击省份，可以看到某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：

构建而成的知识图谱可以清楚的表达想要表达的知识，可以便捷地查找想要获得的信息。

然而点开一个节点后，会显示所有与该节点相连的信息。由于我们的数据量巨大，会出现页面卡顿反应迟缓的问题。所以我们组在演示时选择了一个节点点开进行展示。图谱的设计还可以进一步优化。

我们对已有的数据使用也不够全面，可以加入更多的知识来丰富图谱的内容。

由于一个实体只能显示相连实体的内容，所以我们在分数实体中又将学校名称、专业等信息以属性的方式呈现，有些信息冗余。但能够全面清晰的显示我们所需的信息。

六、实验总结与心得体会

本次实验成功完成了预定目标，能够用知识图谱清晰得表达招生信息。同时在实践过程中也发现对图谱的不同设计也会产生不一样的效果。neo4j 的使用方便了图谱的构建，处理好数据并创建 csv 进行导入就能获得图谱的效果，使得我们能够掌握 neo4j 这一新工具。在实践过程中也遇到了一些问题，如全局变量冲突等问题，最后都得到了解决，也增强了我们解决问题的能力。

在一个项目的综合开发过程中，团队的配合也尤为重要。有了清洗好的数据后，才能顺利的构建知识图谱。有了前端的支持，才能将图谱得以展示。这需要每个人按时按标准的完成自己本应完成的任务，所以也需要组员之间的相互 push。

个人还应给自己提出高标准，降低出错率，以精益求精的精神对待自己的工作，也是对组员的负责，促进项目的顺利进行。

在老师的教导下，我了解了软件开发的流程并加以实践，收获颇多。相信本门课程的学习对以后的工作及科研都是大有裨益的。

2020 年 9 月制