《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 A 组						
姓　　名	白劭宸	专业	人工智能	班级	091181	学号	09118110
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：随着中国教育不断发展，高考已经成为国家选拔人才的公平公正平台。然而，在当今资讯爆炸的时代，如何合理的搜集有效数据并且做出理性决策，已然成为考生和家长心中的一大负担。

目的：在搜集各省市各学校的分段数据基础上，通过内置的智能模型，为考生填报志愿提供理性的分析和推荐。

二、小组任务和个人任务

小组任务：将搜集的各数据按照知识图谱一定的数据模式进行结构化重构，以更加语义可理解的方式提供数据的查询和可视化。

个人任务：整合知识消歧后的数据，在此基础上，设计数据模式，使之在结构上更加合理有效和易读。同时，将数据进一步转化成 Neo4j 可读的规范化数据，为小组下一步输入数据库并进行可视化操作提供前期支持。

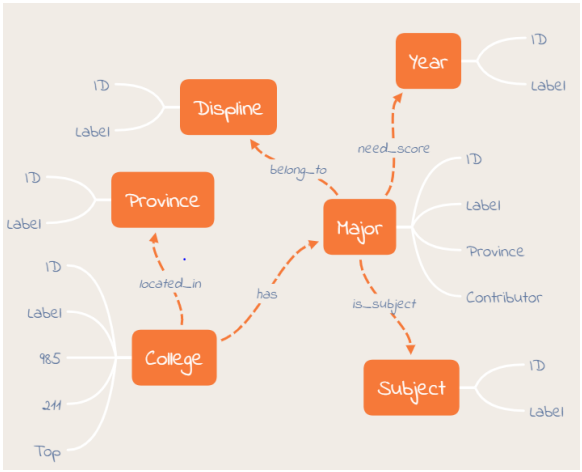
三、个人任务需求分析

消歧后的数据在本质上还是以表为基础的关系型数据库模式，因此在真正加载数据进行图数据库输入之前，还有以下问题亟需解决：

1. **知识图谱的模式设计；**

   将原有的表转化成三元组的模式，需要合理设计哪些数据需要单独设计成实体，哪些需要设计成联系，哪些需要设计成实体的属性等等。

   图谱设计的如下所示，排名信息并未显示，在后续加上。



注： Displine 代表一级学科，Subject 为文理分科

2. 数据转化；

Neo4j 的实体和联系需要统一的全局索引，因此在原有基础上需要将表进行进一步拆分和用索引替代表示。

工作将主要围绕 disambiguated 表展开，表格中包含着大量信息，因此需要在原有基础上进行整理和拆分，并且最后添加索引。

Unnamed:	College	Year	Province	category	Major	score	Contributc	disambigu	ID
0	清华大学	2019	上海	理科	all	615	09118101	all	1304;
1	清华大学	2019	广西	理科	all	660	09118101	all	1304;

disambiguated 表表头

3. 数据清理：

此部分为附加工作。

虽然在前面小组工作中，做了大量清理工作，但是仍然偶有遗漏，需要再一次核实的清理。

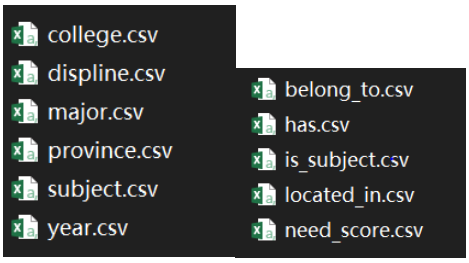
8348	北京理科大学	2019	安徽省	理科	徐特立英才	647	09118107	徐特立英才1403;

如北京理科大学（北京理工大学）

四、实验过程（需附上关键代码及相关说明）

1. 知识图谱设计过程：

根据网上 Neo4j 数据库的数据输入要求，和先前知识图谱模式的设计，我们初步采用 Excel 的内置方法将数据拆分，6 个 entity 表格，和 5 个 relation 表格。



左侧为 entity 表格，右侧为 relation 表格

同时在此基础上，手工对表格进行遍历，对数据清洗的工作做好前期准备，如：哪些属性的格式不符合要求，需要规范；哪些记录出现了歧义错误，需要去替换等等，

2. 数据转化过程：

编写 python 代码，为数据生成全局唯一的索引字段。

```
#province convert
if arr[3].isdigit() == False:
    for pro in province:
        if pro in arr[3]:
            arr[3] = 'p' + str(province.index(pro)+1)
else:
    arr[3] = 'p' + str(arr[3])
```

采用的一般规则为：如果是 college 实体的，在一般索引前面加上’c’以作区分，如果是 province 实体，则加上’p’，以此类推。

要注意的是：由于各数据格式规范性比较差，因此在匹配字段的时候不能采用硬匹配，而应该多采用软匹配（如使用 in，或者是正则表达式匹配）

同时，当一个专业属于多个一级学科时，消歧小组将一级学科以 ；号分割并列于一行当中，这在 Neo4j 的标准里面是不允许的，因此要针对一级学科的属性进行表内的再一次拆分。拆分过程还要注意生成的新记录的全局索引应当是不变的。

```
#disambiguated dealing
if arr[-2].find(';')>=0:
    major = arr[-2]
    ID = arr[-1]

    arr_major = major.split(';')
    arr_ID = ID.split(';')

    for m in range(len(arr_major)):
        arr[-2] = arr_major[m]
        arr[-1] = arr_ID[m]
        writer.writerow(arr)

    flag += 1

else:
    arr[-1] = arr[-1][:-1]
    writer.writerow(arr)
```

3. 数据清洗过程：

此步多用 Excel 的查找和替换实现，但是数据记录太过庞杂，虽然经过人眼筛选，但是仍然不能保证数据的规范，因此最后，在容易出现歧义的属性上进行数据验证器编写是十分必要的。

```
if name == 'entity':
    index = int(arr[0][1:])
    if index != flag:
        return False, flag
    else:
        flag += 1

elif name == 'province':
    province = arr[2][1:]

    if province.isdigit() == False:
        print(flag)
```

验证器不仅能根据一定规范，判断数据是否已经整体符合要求，同时也可以返回出错行，方便查找和改正。

五、实验结果与分析

College:ID	Label	985	211	top	
c10001	北京大学	1	1	1	
c10002	中国人民大	1	1	1	

Major:ID	Label	Province	Contributor
m1	all	p9	09118101高捷
m2	all	p29	09118101高捷

:START_ID	need_score	:END_ID	:TYPE
m1	615	y3	NEED_SCORE
m2	660	y3	NEED_SCORE

注：表格分别为 College(entity)，Major(entity)，need_score(relation)

可以看到，通过表格的拆分，基本上实现了数据库的“一事一地”原则，较为有效的实现了数据的规范和索引的添加。同时满足了 Neo4j 对于数据输入的格式的要求。

同时，在 relation 的创建上也较好的实现了语义表达，赋予每个节点和联系更多的信息。有效的为下面知识图谱的输入做好了前期准备工作。

六、实验总结与心得体会

于我而言，这一次百人编程作业，着重点并不在个人的代码编程能力，更重要的是在组内流程的配合和组间的协调。

在一个人编程的情况下，逻辑是全部由自己掌控的，因此只要顺着思路往下写就行了。但是多人编程，就更需要注重数据输入输出格式的规范，和理清他人代码的逻辑。

看似简单的要求，实现起来并不容易。在项目开始的时候，由于对实现目标和过程还没有完整的认知，是很难提出合理的数据要求的，因此在实践过程中，难免会出现需要自己重新整理数据的情况。在产业化的项目里，这不仅是对效率的极大浪费，也会威胁项目的最终成果展示。但是考虑到现在大家还在学习，更多时候，都是保持着对他人体谅的心情，自己默默赶工的。因为也难保下一组对你也是如此的心情。

在这样别扭而磕磕绊绊的实现过程中，我深刻体会到了两样东西的重要性：数据和注释。因为在小组作业中，组内传递的往往不是代码而是数据，因此在移交成果的时候，能清晰明确的描述自己的数据结构和内在逻辑，对于下一组代码的编写是由莫大帮助的。同时，在数据量较大的表格中，应当适当别写数据验证器，保证数据的完整。在代码的基础上添加注释，除了帮助自己理顺思路以外，也是在给阅读代码的人提供遍历。

总之，编程脱离了单打独斗，考验的就不仅仅是最后的成果和效率那么简单的事情了，代码和任务的处理是否有逻辑，都一一成了考察目标。结果就是，代码更加的注重“过程”这样的指标。